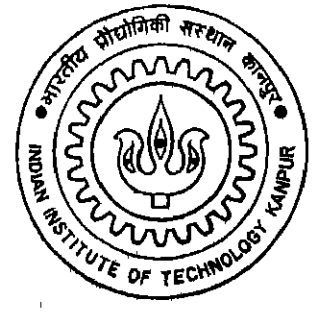
Implementation of Multicast Backbone on IITK Campus LAN

by
SANDHYA V. SULE



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
DECEMBER, 1994

Implementation of Multicast Backbone on IITK Campus LAN

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of
Master of Technology,
by

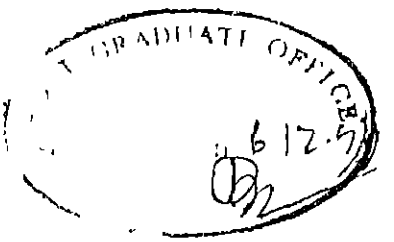
Sandhya V. Sule

Department of Electrical Engineering
Indian Institute of Technology Kanpur
December 1994

Abstract

Applications such as multi-media multi-source information services, desk-top conferencing, joint editing and preparation of documents etc., allow users of a network to participate in co-operative work across the network These applications require the underlying network to provide group communication features such as multicast addressing and group management. Further, to use the available network bandwidth efficiently, it is essential that the multicast routing algorithms produce low-delay routes In this project, the implementation aspects of providing group communication support have been studied The IETF-MBONE experiment has been emulated to set up a virtual network over parts of the IIT Kanpur campus LAN. This virtual network forms an IP multicasting backbone (MBONE) to route multicast packets among hosts on different subnets in the network. Three machines on three different subnets in the Campus network have been configured as multicast routers. These routers route multicast packets to/from hosts on these three subnets. These machines also provide operating system support for sending and receiving IP multicast packets Two additional machines have been configured to support only sending of multicast packets. For configuration of MBONE, public domain software available over the Internet has been customized for the Campus network. The proper functioning of the campus multicast backbone has been demonstrated using a public domain software which initiates and manages multicast sessions

Certificate



This is to certify that this thesis titled **Implementation of Multicast Backbone on IITK Campus LAN** has been carried out satisfactorily by **Sandhya V. Sule, (Roll No : 9210438)** as partial fulfilment for the degree of *Master of Technology*, under our supervision and that it has not been submitted elsewhere for a degree.

K. R. Srivathsan
Professor
Department of Electrical Engineering
IIT Kanpur

Vishwanath Sinha
Professor
Department of Electrical Engineering
IIT Kanpur

Acknowledgements

I gratefully acknowledge Prof V. Sinha for giving me the opportunity to do Masters in Telecommunication Networks, with sponsorship from the Nationally Co-ordinated project on Telematics His unwavering support during my M. Tech program has enabled me to fruitfully complete it, inspite of several unforseen difficulties I take this opportunity to also thank Prof K. R. Srivathsan for his continued guidance throughout the project The impromptu discussions with him, have motivated me to work in the area of communication networks. I would also like to acknowledge Dr. Dheeraj Sanghi who introduced me formally to the world of networks He has been closely associated with this work and I am very grateful to him for his help. I wish to thank Dr. P. K. Kalra, Prof. Palaniswamy and Prof. M U Siddiqi, for extending the use of their machines for my project. Without their co-operation, this project could not have been completed

Thanks are also due to my friends Sanjeev, T S Rao, Navpreet, Vivek, and Sujata who have given me all the help that I asked for Special thanks are due to Padmaja and Arvind

Last but not the least is the contribution made by my family. While doing my M. Tech , I have very often neglected my home, but my husband Viren, and son Shantanu have shown great understanding and encouraged me always.

Contents

List of Figures

List of Abreviations

DARPA Defense Advanced Research Projects Agency

DVMRP Distance Vector Multicast Routing Protocol

ERNET . Educational and Research NETwork

ICMP Internet Control Message Protocol

IGMP . Internet Group Management Protocol

IP . Internet Protocol

LAN . Local Area Network

MBONE · Multicast BackBONE

MOSPF : Multicast Open Shortest Path First

OSPF Open Shortest Path First

RPF · Reverse Path Forwarding

TCP . Transmission Control Protocol

TRPB Truncated Reverse Path Broadcasting

WAN · Wide Area Network

Chapter 1

Introduction

A computer network is an interconnected collection of several autonomous computers. This network may be a small local area network (LAN), or a large network of interconnected LANs. An application program executing on one machine in the network may interoperate with applications executing across the network to provide a service varying from simple ASCII file transfer to the more complex service such as multi-media document retrieval and transfer. The network architecture, should, therefore provide support for distributed applications, network management and maintain satisfactory performance. Further, services provided by the underlying network must be engineered for efficient execution of the network applications.

1.1 Traditional Network Applications

Different applications require different services from the underlying network. The following are typical examples .

1. Applications such as information services, directory services, library services, etc., require *remote access to databases*

2. Airline reservation systems, inventory control systems, teller systems, commercial credit card systems etc., require *remote updation* of databases, in addition to *remote access* and *transactions processing*

3. Electronic mail between users on the network requires that messages should be sent, received, processed and forwarded. Usually, this store-and-forward service is implemented over the usual network services.

4. Applications requiring distributed computing power or load sharing need the network to perform the assigned task within some predefined maximum delay

1.2 Emerging Network Applications

With the availability of very high speed communication links using optical fibre technology, high power workstations and efficient VLSI implementable algorithms for encoding and decoding of various media like video and voice, there is an emerging set of applications which can be supported on a data network Some of these are Multimedia multi-source information services, integrated voice and graphic mail, desktop multimedia conferencing, network based automation in industrial applications such as process control, joint editing and preparation of documents over the network, on demand video, and telemedicine The RAPPORT system [1] and the Multimedia Conferencing project [11] are two examples which focus on the networking requirements and user interface development for real time conferencing applications

1.3 Network Support for Co-operative Work

Common to this emerging class of applications is the requirement for the network to support co-operative work between remote users This support, provides an efficient method by which remote users can interact within a group.

The Figure 1 1 illustrates a typical classification of co-operative work support features over a network as given by Sakata [22] According to this classification, the co-operative work-support features can be divided into two broad classes:

1. those allowing shared workspace, such as audio, video communication support and group co-ordination, and

2 those allowing sharing of the information space viz integrated support for personal and group work, and interactive data exchange and handling within the group

In case of applications such as multi-source information services, the latter class of features is important, whereas, in case of applications like multi-media conferencing, both classes of features should be provided by the group communication service on the underlying network.

In a network of interconnected LANs, hosts on the network can participate in co-operative work if the network provides a low level service with some basic group communication features such as group addressing and group management. The concept of grouping processes has traditionally been used in a distributed systems environment [3] to achieve goals such as parallel processing, increased data availabilty, shared resources, reduced response times, and reliablility. Similarly, in distributed applications such as conferencing, group multicast or group addressing reduces transmission overheads on the sender

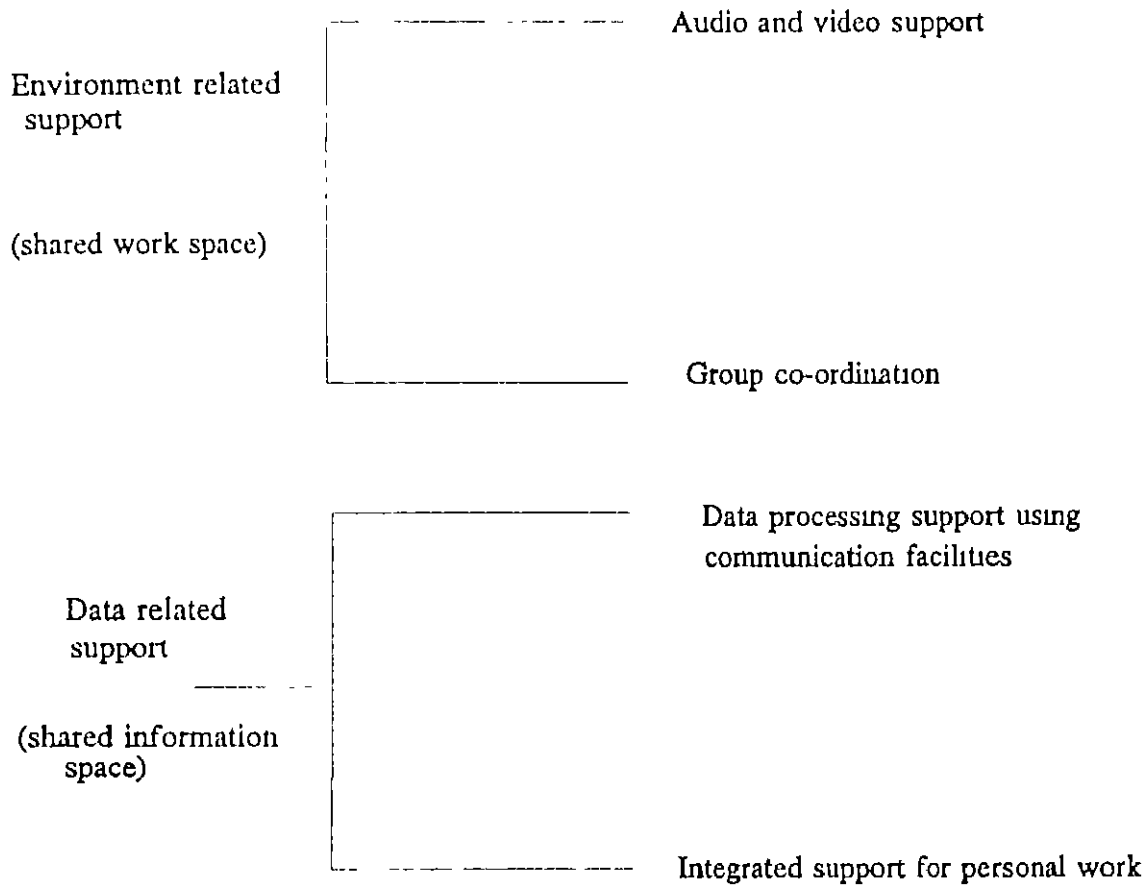


Figure 1.1: Co-operative Work-support Features

and allows a host to send information to other hosts in the group, whose address is unknown or changeable. This concept of *group addressing* is widely used in LANs.

1.4 Group Communications Over Networks

To illustrate typical group communication features, a network of interconnected LANs is considered as shown in the Figure 1.2.

Hosts A, B, C, D, and E are located on LANs 1, 2, 3, 4, 5 respectively. Now, consider the following scenario :

- Hosts B, C, D, and E are members of group *g1*.
- Host A does not belong to group *g1* but desires to send packets to *g1*.

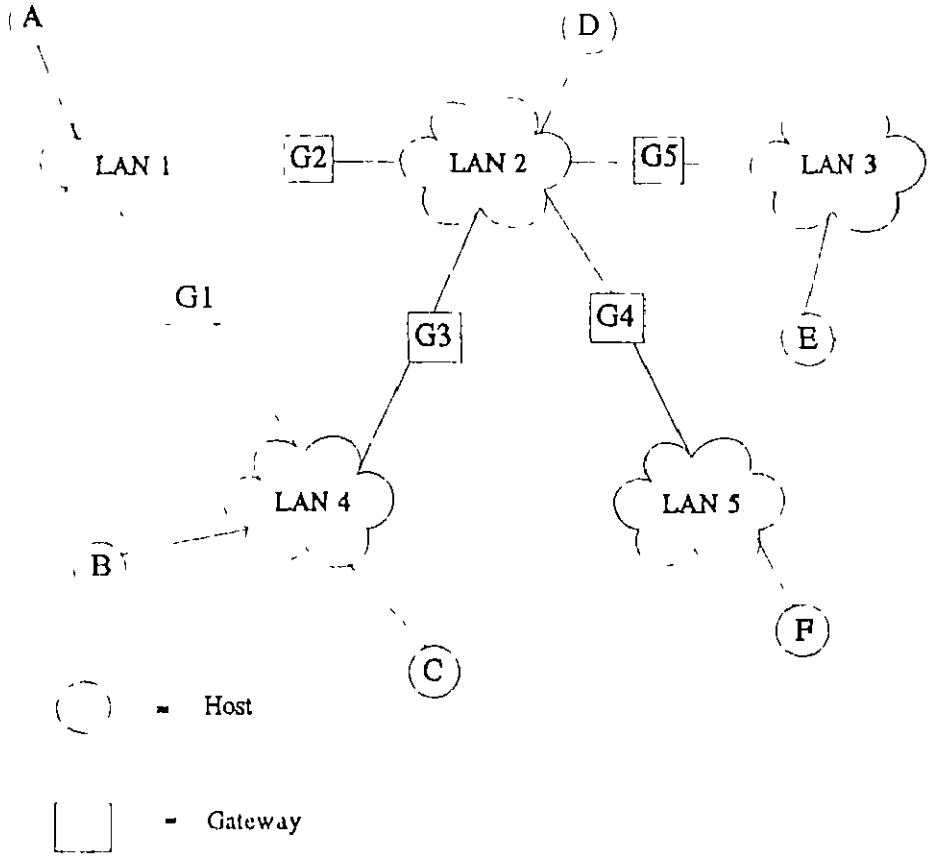


Figure 1 2 A Typical Network of Interconnected LANs

In order that host A can send packets to group *g1*, and for hosts B, C, D, and E, can receive these group addressed packets, the following requirements must be met

- Host A must be able to send packets with destination address of the packet as *g1* Hosts B, C, D, and E should be capable of recognizing the packets with the destination address *g1* as their own The concept of sending packets to a pre-defined group of hosts is called **multicasting.**
- A host can receive packets if it belongs to a group Therefore, a mechanism for subscribing and unsubscribing to a group should be available within every host that wants to participate in co-operative work over the network. Also, each of the gateways G1, G2, G3, G4, and G5 must know of the groups on each of its links to forward the group addressed packets to all members of the group. Such group management functions

need to be provided by a group management service executing in both the hosts and the routers

- Finally, it is desirable that multicast packets must be delivered to all *members of the group with minimum delay*. This is crucial in applications such as conferencing and resource location. The delay properties of large internetworks are affected by the greater geographic extent and large number of links and switches. However, the use of high speed links and low-delay long distance communication links can significantly reduce the delay. It is therefore essential that **multicast routing algorithms** produce low delay routes.

One of the most widely used Internetworking technology is the **DARPA TCP/IP Internet** [7]. The TCP/IP Internet provides a set of network standards which can be used to communicate across any set of interconnected networks. The next section gives a brief overview of the TCP/IP Internet to understand how some basic group communication services can be provided within this network architecture.

1.5 The TCP/IP Internet - An Overview

The TCP/IP Internet is an unified interconnection of independent networks providing three sets of services, as shown in the Figure 1.3. At the lowest level, it provides a connectionless datagram delivery service. At the next level, it provides a reliable transport service. At the highest level application services such as *remote login*, *file transfer* and *e-mail* are provided.

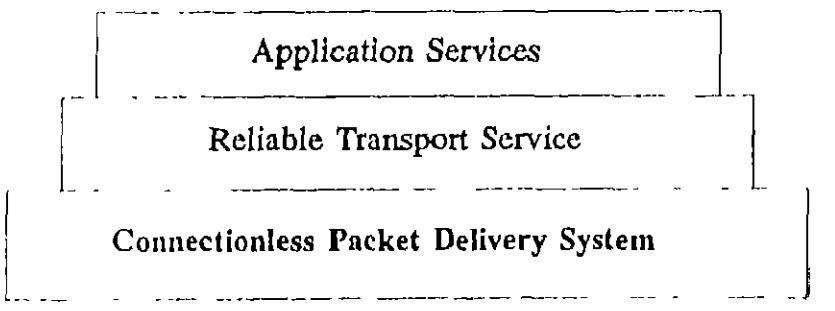


Figure 1.3 The three components of the Internet Services

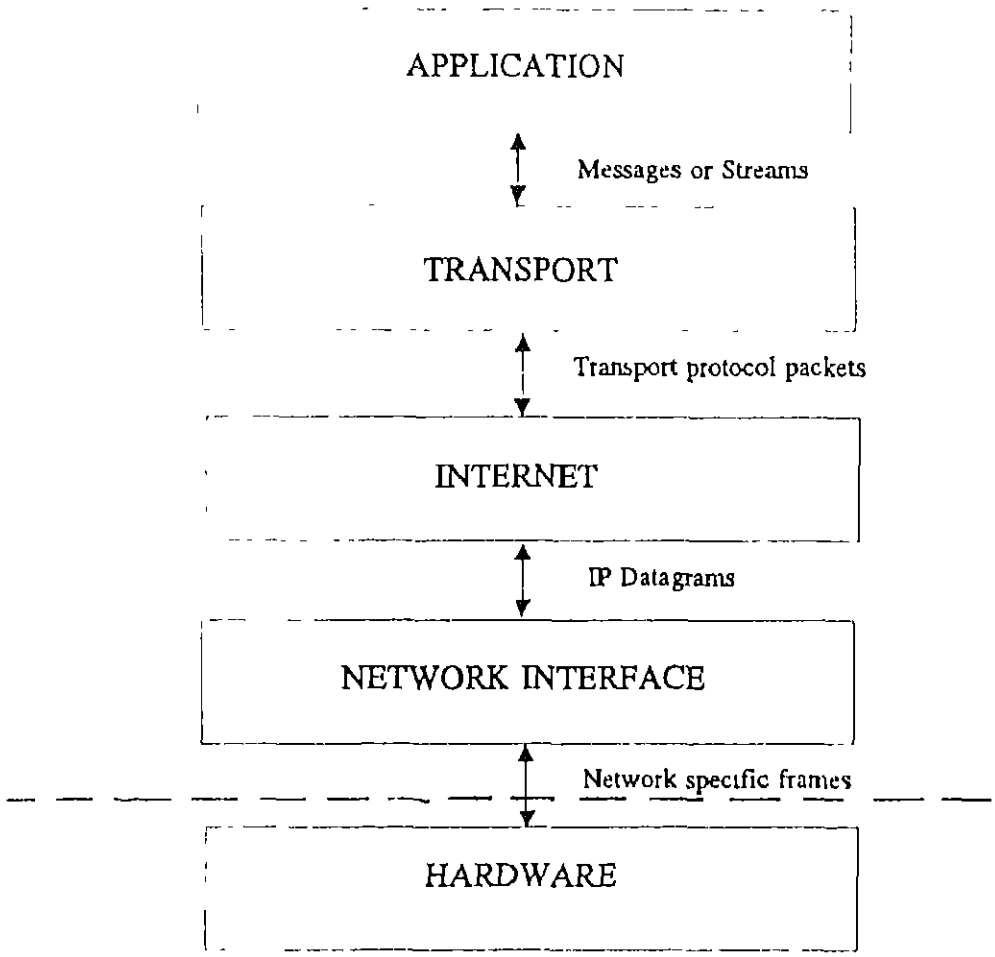


Figure 1.4: The four conceptual layers of TCP/IP

The Figure 1.4 shows the Internet layering. This figure also shows the objects passed between layers. The network interface layer is synonymous to data link layer as defined in the OSI protocol stack. The network layer is concerned with issues such as the exact format of Internet datagrams, host and network addressing, and the routing of datagrams across the Internet. This layer also defines a connectionless datagram delivery service. The transport layer protocols provide end-to-end connections between application processes executing across the network.

An unreliable datagram service at the transport level provides a connectionless communication among the applications, whereas the Transmission Control Protocol (TCP) provides a reliable, flow-controlled, stream service between two applications. Figure 1.5 shows the dependencies among major higher level TCP/IP protocols. The details on Internetworking with the TCP/IP are given in [7]. In the TCP/IP protocol suite, the Internet layer

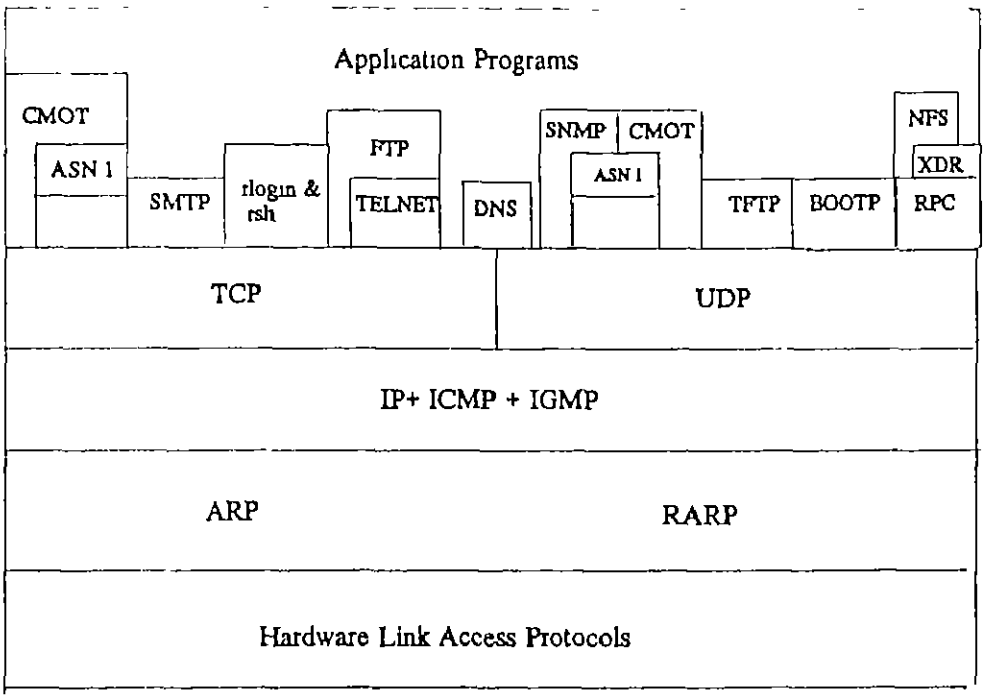


Figure 1 5. Dependencies among the major TCP/IP Protocols

[14] performs the addressing and routing functions. Thus group communication services such as group addressing, group management and routing are extensions to the Internet layer

1.6 Objectives Of This Project

As discussed above, applications such as multi-source multi-media information services, desktop conferencing and joint preparation of documents need group communication support services on a network. Till recently, there were no standards specified for group communication services in the TCP/IP Internet, although, a standard for group management, called the Internet Group Management Protocol (IGMP) did exist [16]. However, due to increasing infrastructure support such as high speed links, and switches etc., efforts towards implementing experimental protocols for multicasting, multicast routing and group management have gained momentum since the early '90s. Presently a standard exists for the extensions to IP for multicasting and IGMP [19] but

yet there is no standard for multicast routing

Some of the first efforts towards developing group communications services over the Internet were made by the Internet Engineering Task Force (IETF). The IETF carried out the **Audiocast** experiment [23] to transmit live audio and video from the IETF meeting site to destinations across the world. A fallout of this experiment was the Multicast Backbone or the MBONE setup across various Internet sites over the world. The MBONE idea is to create a virtual network layered on top of portions of the physical Internet to support routing of the IP multicast packets. In the MBONE experiment, IGMP was used for group management and a distance vector style multicast routing algorithm (DVMRP) was implemented for routing of the IP multicast packets.

In this project, the enchancements of a network site to provide host multicast, group management and multicast routing have been studied. The IITK campus network has an extented LAN topology with Internet TCP/IP implementation An experimental Multicasting Backbone has been setup to provide a primitive level of group communication service over the IITK network, based on the IETF MBONE experiment The public domain software available over the Internet has been used for this purpose. This project follows the Internet architecture in totality.

1.7 Organisation of This Thesis

The rest of this thesis is organised as follows :

In Chapter 2, details about properties of group communication and group management on the Internet have been described. This chapter also describes how hosts on the Internet can send/receive multicast packets to/from a predefined group

The routing and forwarding algorithm for the multicast packets as standardised in the Internet is explained in Chapter 3.

In Chapter 4, the details of the enhancements on the campus network for multicasting, routing and group management are given.

Chapter 5 concludes with scope for further work

Chapter 2

Group Communications Over Internet

As seen in chapter 1, at the lowest level, group communication services provided by a network deal with issues such as group addressing, group management and the efficient routing of the group addressed packets. The Internet Authorities Board (IAB) provides a standard [19] for IP host multicasting (or group addressing) and group management. These protocols have been developed over a predefined group communication model. In this chapter, we first study the parameters which influence the design of the group communication protocol supported by a network. The Internet group communication model is presented next In the following section, the features of the Internet standard for host IP multicasting are reviewed along with an overview of the Internet Group Management Protocol (IGMP), which is used to carry routing updates and group membership reports in the IP multicast implementation.

2.1 The Group Communication Model

Protocol design for group communications is based on a pre-defined group communication model. This model is defined by parameters such as the group types, group characteristics, packet delivery characteristics and the underlying network environment. These parameters are, in turn, defined by the applications which need to use the features of group communications The various parameters on which the group communication model depends are given in the Tables 2 1 and 2.2 A typical example of this would be the group communication model for the Internet.

2.1.1 Internet group communication model

In the Internet, the group communication features are designed to support distributed applications across the Internet. S.Deering and E Cheriton [10] have defined a model which has been adopted for the design of the IP multi-

casting and multicast routing in the Internet This model specifies the types of groups, the group characteristics, the datagram delivery characteristics and the underlying communication environment

Group Type	features	Usage
Local	all members of the group lie on the same LAN or Administrative Domain.	Distributed systems
Sparse	no. of members are few and belong to seperate Administrative Domains.	Conferencing
Transient	formed only at the time of usage	"
Permanent	Long lived	news
Pervasive	Large no. of members, atleast one member per group on each LAN	Directory Server.

Table 2.1 Types of groups.

group character.	features	usage
open	sender need not be a member of the group	directory servers,
closed	sender must be a member of the group	conferencing news, name
static	membership remains undisturbed over long intervals of time	name servers
dynamic	group members can join or leave the group at will witnout negotiations with other members of the group.	conferencing

Table 2.2 Group Characteristics.

In this model, communication is provided by multi-access networks (LANs and possibly satellite networks) interconnected by packet switching nodes in an arbitrary topology A typical example is shown in the Figure 2 1

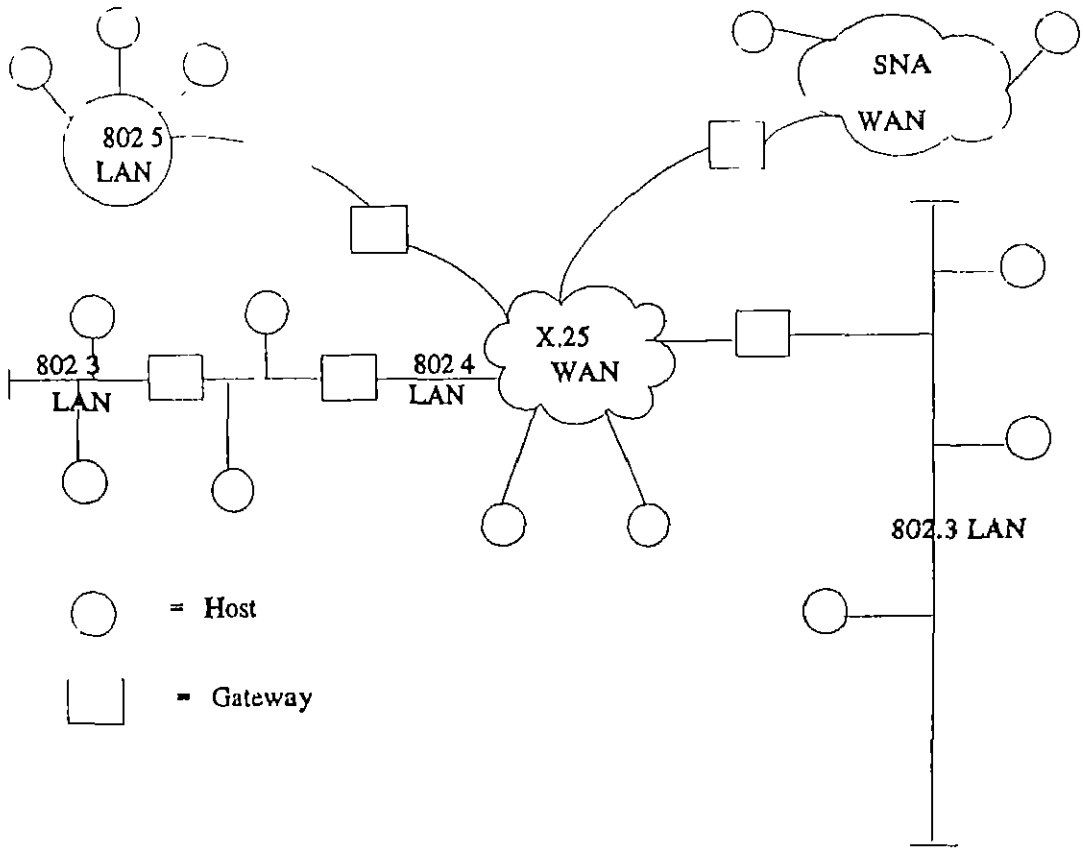


Figure 2.1· A typical Internet toplopy

Point-to-point links may provide additional connections between the switching nodes or from switching nodes to isolated hosts, but almost all hosts are directly connected to LANs. The LANs are assumed to support local network multicasting The supported group type is **open** and its characteristics are **dynamic** There is no limit on the number of members in a group All group addresses are allocated when the group is initiated, except for a few pre-assigned permanent ones and the addresses are released when the group is dissolved. Thus, a group exists as long as there is atleast one member in the group, except in the case of a few well known permanent groups. The permanent groups have well known administratively assigned IP multicast addresses.

When implementing the IP multicast, the designers have assumed that the multicast packet is delivered with the same **best effort delivery** [7] as the IP unicast datagrams This implies that packet loss should be handled by the upper layers The delivery delay is assumed to be minimized with the implementation of multicast routing algorithms which produce low delay routes Since the IP multicast addressing and routing are extensions of Internet layer, the Internet protocol which provides the connectionless datagram delivery mechanism, is first reviewed below

2.2 Internet Protocol

The Internet Protocol (IP) provides three basic definitions

- IP defines the basic unit of data transfer used throughout the Internet, which is called as the IP datagram Thus, IP specifies the exact format of all data as it passes across a TCP/IP Internet.
- IP software performs the routing function, choosing a path over which data will be sent
- IP includes a set of rules that embody the idea of unreliable packet delivery. The rules characterize how hosts and gateways should process packets, how and when error messages should be generated, and when packets can be discarded.

2.2.1 The Internet datagram

The Internet datagram is divided into header and data areas The header contains the source and destination IP address, and additional type fields that identify the contents and characteristics of the datagram. The datagram header is shown in the Figure .2.2. Because the datagram processing occurs in software, the contents and format are not constrained by any hardware. The significance of each field of the datagram header is given in [7]. In the IP multicast extensions only a few fields of the IP header are of significance.

They are .

- **Time To Live** 8 bits .

  The time to live specifies how long, in seconds, the datagram is allowed to remain in the Internet. Gateways and hosts which process the IP datagrams must decrement the time to live field as time passes and discard the packet when the time to live expires. Since estimating the lapsed time is difficult, a simple rule is followed at every gateway along the path from the source to the destination :

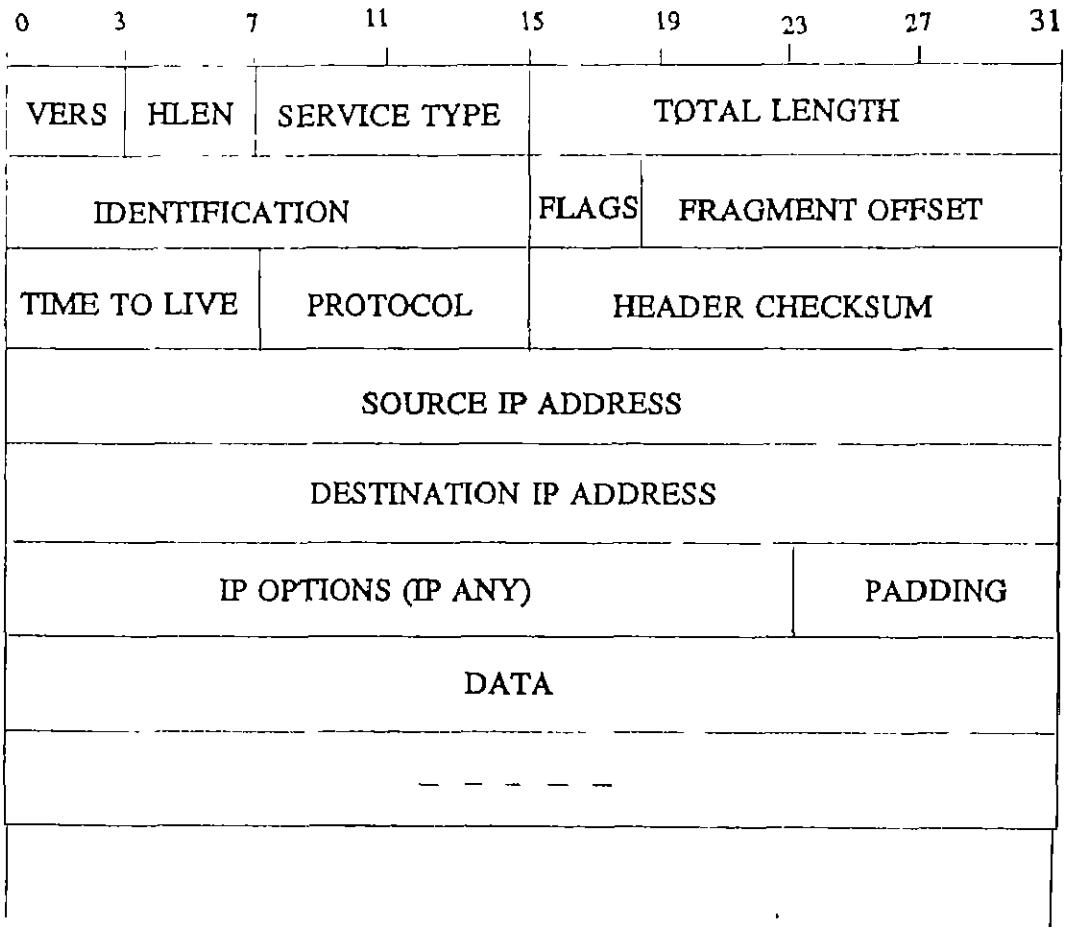


Figure 2 2. Format of an Internet Datagram

```
IF (IP-ttl >= 1)
   decrement IP-ttl
   forward the packet to the next-hop gateway
   in the path to destination
ELSE
   discard it.
```

This field has been used to limit the scope of travel of a multicast datagram using a threshold mechanism. This is explained later in the chapter.

- **Source IP Address** : 32 bits ·
- **Destination IP Address** : 32 bits .

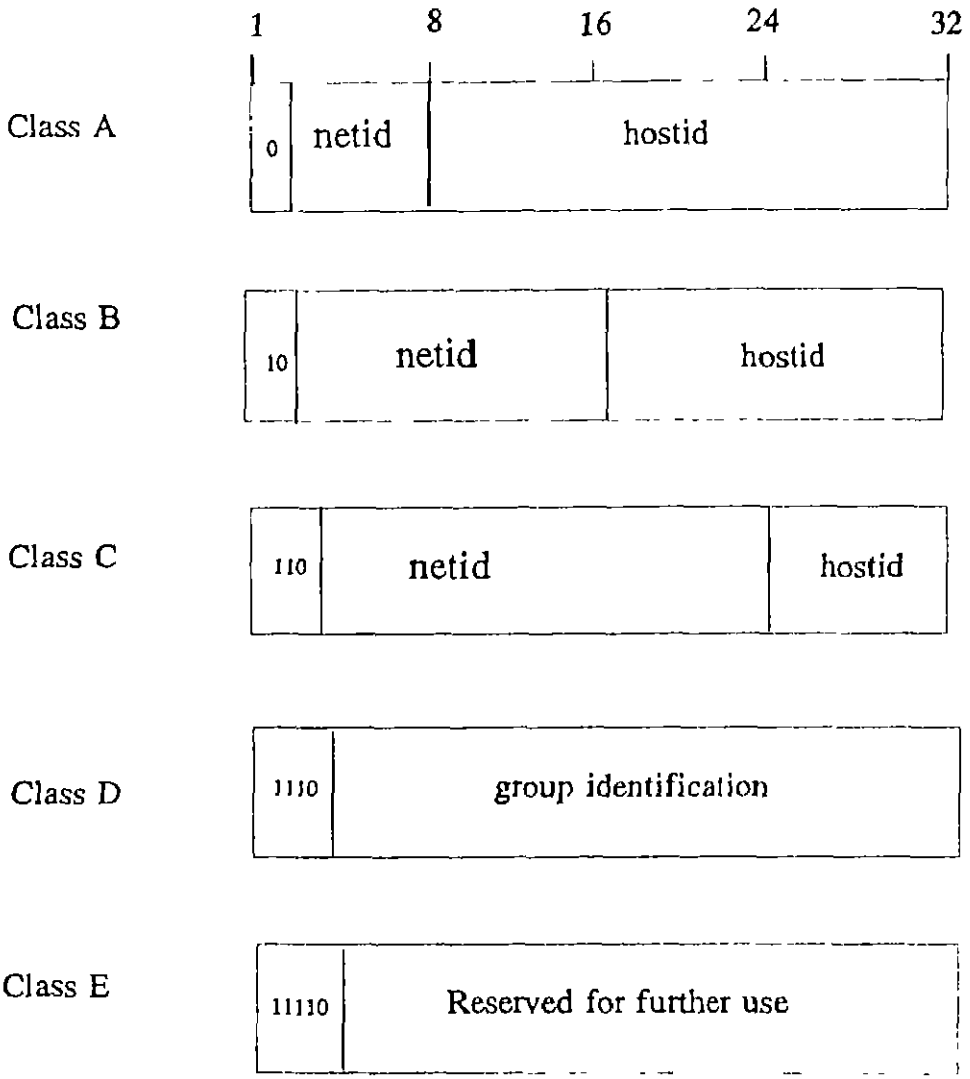


Figure 2.3 Forms of Internet Addresses

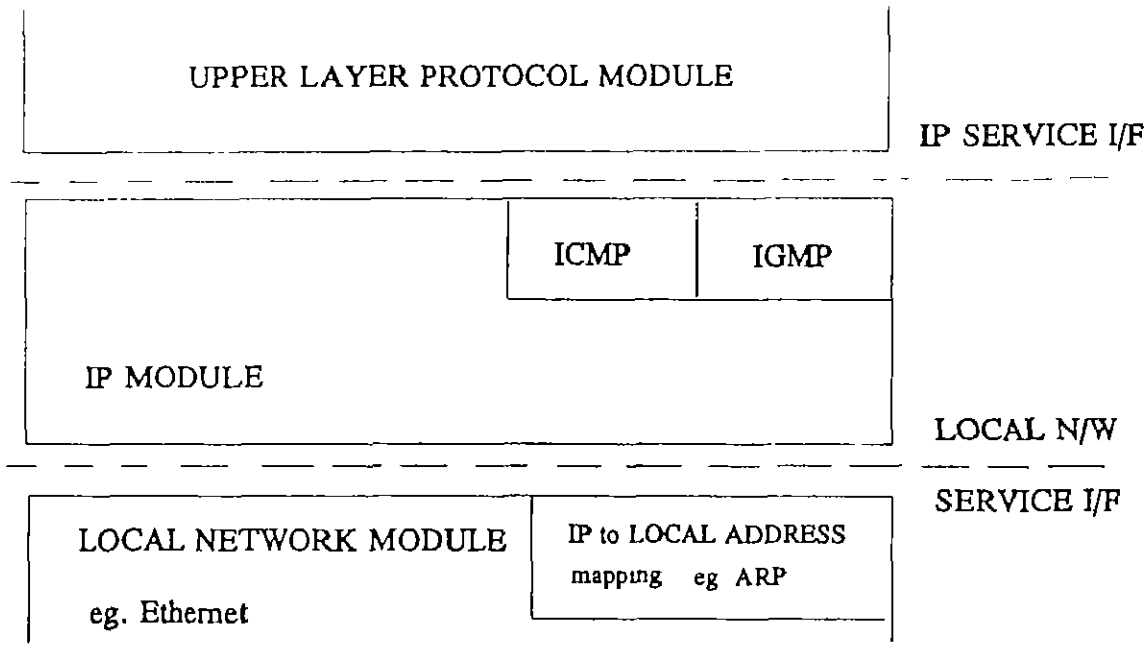


Figure 2 4: Host IP Implementation - Layered Model

- On those networks that are up and reachable within the IP time-to-live of the multicast datagram, the attached multicast router completes the datagram delivery by transmitting the datagram as a local multicast

From the above, the following implementation issues arise:

1. How can hosts send IP multicast datagrams?
2. How can hosts receive IP multicast datagrams?
3. How do routers know of the group memberships on its links?

This standard addresses the above issues on the basis of the host IP implementation layered model shown in the Figure 2 4

In this model, ICMP and IGMP are assumed to be implemented for hosts which confirm to the level 2 IP multicast. For the level 1 hosts, the host IP implementation should only support the transmission of the IP multicast datagrams At the lowest level in these extensions is the mapping of the IP multicast address to the link level address (e g. IP to Ethernet address mapping). In the following section the multicast address resolution problem is discussed.

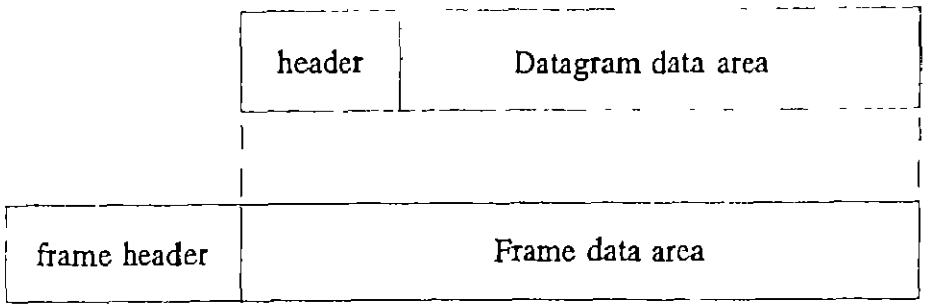


Figure 2 5 The encapsulation of IP datagram into a frame

2.3.1 Resolving the multicast address

When datagrams move from one machine to another, they must be transported by the underlying physical network The network hardware does not understand the datagram format but merely encapsulates the datagram by adding an extra frame header The frame header contains besides other fields, the physical (hardware) addresses of the source and destination host machines Thus, as shown in the Figure 2 5, the entire datagram travels in the data portion of the network frame

The mapping of the Internet address to the network specific physical address is handled by low level protocol such as ARP [7]

IP multicast addressing

In the IP, the datagram's destination address specifies the recipient of the datagram For multicasting a datagram, the destination IP address is a Class D IP multicast address of the form :

1110	Group Identification

The first four bits identify the address as multicast. The remaining 28 bits identify a particular **host group**. The group field does not identify the origin of the group and does not contain any network address. When expressed in dotted decimal notation, the multicast addresses range from :

224.0.0.0 to 239.255 255 255

The address 224 0.0.0 cannot be assigned to any group The address 224 0 0 1 has been permanently assigned to the **all_hosts_group** and is used to reach all hosts and gateways capable of multicast on a local network. The address 224 0 0 4 is used to address only the *routers* participating in the multicast. All the routing queries and updates are sent to this address

An IP multicast address can only be used as destination address It cannot be used as a source address or in the source route. Also, no Internet Control Messages (ICMP) [7] such as destination unreachable or time exceeded can be generated about multicast datagrams

The mapping of the IP address to the link level address is done by the Local Network module (refer Figure 2.4). To understand the mapping, the Ethernet local network module is considered

Ethernet multicast addressing

The Ethernet directly supports the sending of local multicast packets by allowing multicast addresses in the destination field of the Ethernet frames. An Ethernet multicast address is identified by the use of low-order bit of the high order octet in the six octet Ethernet address. In dotted decimal, this address is identified by

01 00 00 00.00 00

The Ethernet hardware can be configured to accept packets for a small set of multicast addresses besides its assigned physical (unicast) address and the broadcast

Mapping IP address to Ethernet address

The standard specifies the mapping of the IP multicast address to the Ethernet address as follows:

> Place the low-order 23 bits of the IP multicast address into the low-order 23 bits of the Ethernet multicast address 01.00.5e.00.-00.00

For example, the IP multicast address 224.0 2 1 is mapped to 01.00.5e 00.-02.01 Ethernet address.

This mapping is not unique. Because IP multicast addresses have 28 significant bits that identify the multicast group, more than one group may map onto the same Ethernet multicast address. This scheme has been chosen as a compromise, by the designers On the one hand, using 23 bits of the 28 bits for a hardware address means most of the multicast address is included. This set is large enough so the chances of two groups choosing addresses with all low-order bits identical is very small. On the other hand, arranging for IP to use the fixed part of the Ethernet multicast address space makes debugging easier and eliminates interference between IP and other protocols that share an Ethernet. To avoid receiving packets not destined for that hosts, the IP software should carefully check addresses of all incoming datagrams and discard the unwanted ones.

Unicast vs Multicast address resolution mechanism

Address resolution in the case of unicast datagram is done using the Address Resolution Protocol (ARP) [7]

In case of multicast IP datagrams, the destination address specifies a group of hosts The multicast physical address (for eg , Ethernet) depends only on the group address The address mapping as described for Ethernet allows hosts' network interfaces to transport multicast IP packets across the network. This mechanism is static when compared to the dynamic binding provided in case of the unicast datagram transmission, in the sense that a **change in the physical address of the destination host does not alter the destination physical address of the transmitted datagram** The receiving host should only alter its hardware address filter to recognise that physical address as its own.

2.3.2 Extensions for sending IP multicast datagrams

For sending an IP datagram to a particular IP destination address, the upper layer protocol module specifies the IP destination address to be filled in the destination address of the IP datagram header. This destination address may be unicast or multicast. The modifications required to send the multicast datagram are summarized below, with reference to Figure 2 4 The complete details are available in [19].

In case of unicast datagrams, the IP service interface invokes the **SendIP** operation to send IP datagrams. In case of the multicast datagrams the same **SendIP** operation can be used, with the destination address as the desired IP multicast address (class D) This interface should additionally provide options that can be specified by the upper layer protocol, when submitting the datagram for transfer across the network, for

- specifying the TTL value in the IP datagram header,
- the network interface on which the datagram should be sent and
- whether the datagram should be looped back to the sending host.

To support the multicast IP datagrams, the IP module should be able to recognize IP host group addresses when routing outgoing datagrams. For routing, the following logic should be implemented .

```
IF IP-destination is on the same local network,
OR IP-destination is a host group
send datagram locally to the IP-destination
ELSE
   send datagram locally to GatewayTo( IP-destination )
```

If the sending host is also a member of the destination-group, a copy of the datagram must be looped back unless inhibited by the upper layer protocol.

No change to the local network interface is necessary. This interface only needs to support the destination address as a Class D address when it invokes the SendLocal operation to pass the datagram to the local network module.

For networks that directly support multicasting, such as Ethernet or other network topologies conforming to the IEEE 802.2 standard, the local network module should incorporate the mapping function for mapping the IP multicast address to the Local network multicast address. This mapping has already been explained for the Ethernet, in Section 2.3.1. Details of mapping in case of networks not supporting hardware multicast are given in [19].

2.3.3 Extensions for receiving IP multicast datagrams

For an upper-layer protocol module to receive multicast datagrams addressed to a particular group, it must first be a member of that group. Thus, extensions to the IP for receiving multicast datagrams essentially requires that, it provides some mechanism by which the upper-layer protocol modules can subscribe or un-subscribe to a hosts group, identified by a Class D IP address.

The IP Service Interface must be extended to provide two new operations:

```
JoinHostGroup(group_address , interface)
LeaveHostGroup(group address , interface)
```

Hosts could join the same group on more than one interface. It should also be possible for more than one upper-layer protocol modules to request membership of the same group.

To support the reception of the multicast IP datagrams, the IP module should maintain a list of the host group memberships associated with each network interface. An incoming datagram destined to one of those groups should be processed exactly the same way as datagrams destined to one the host's individual addresses. To correctly handle the incoming datagrams, the following logic should be implemented.

```
IF IP-destination group address is one of ours
   and it arrived on the network interface on which
   this host is a member
   accept  the multicast IP-datagram
ELSE
   discard the datagram.
```

The list of the host group memberships should be updated in response to the JoinHostGroup and LeaveHostGroup requests from the upper-layer protocols. The IP module must also be extended to implement the IGMP protocol as specified in [19] A brief summary of this protocol is given in the next section

The local network interface should deliver the incoming local network multicast packets to the IP module using the same **ReceiveLocal** operation as in the case of unicast packets Additionally, this interface should provide two new operations for the IP module to specify which multicast packets the local network module should accept.

These are

```
JoinLocalGroup( group-address )
LeavelocalGroup( group-address )
```

To support the reception of IP multicast datagrams, the local network module must set the address filter of the network hardware interface, to recognize the desired multicast addresses. For interfaces with inadequate filtering capability, the address filtering should be done within the local network interface software This is not mandatory For networks supporting only pure broadcast, the packets have to be accepted and passed to the IP module for IP-level filtering.

2.4 Internet Group Management Protocol

Similar to the ICMP in its relationship to IP, the Internet Group Management Protocol or IGMP is the TCP/IP standard for group mangement in the Internet The IGMP is used by hosts on the Internet to communicate group membership information with the gateways and by the gateways to communicate the group information among themselves. Like ICMP, IGMP messages are also carried as data portion of the IP datagrams and it forms an integral part of IP. The Figure 2.4, illustrates this. IGMP is therefore required to be implemented on all machines (hosts and routers) that participate in the IP multicast. The overview and implementation are outlined in the standard documented by **S. Deering** [19]. This has been included in Appendix A for the sake of completeness. The use of this protocol is illustrated in the next chapter.

Chapter 3

IP Multicast Routing

In Chapter 2, the host capability for multicasting was studied. The other major enhancement needed at the network level to allow hosts to execute group based applications, is the routing of the multicast packets generated by these applications Deering [9] has proposed several algorithms for routing of multicast datagrams in internetworks and extended LANs The goal is to design multicast routing algorithms, which produce low-delay routes across the network. These algorithms have been proposed as extensions to different styles of unicast routing, such as the distance-vector and link-state routing, commonly used in internetworks. There is as yet no standard for multicast routing in Internet Experimental implementations are available for the distance-vector style multicast routing (DVMRP) [18] and link-state multicast routing (MOSPF) MOSPF is a multicast routing protocol derived from the Open Shortest Path First (OSPF) [20] routing protocol In this project we have used the experimental implementation based on the distance-vector multicast routing (DVMRP), which is described in detail below The next generation multicast routers would probably implement the MOSPF protocol due to the use of the OSPF algorithm as the new Interior Gateway Protocol (IGP) instead of the present IGP, which is a distance vector style protocol. The DVMRP as mentioned earlier, is derived from Routing Information Protocol (RIP) [17]. RIP is briefly described in the next section for better understanding of the DVMRP

3.1 Routing Information Protocol - Overview

As described by Hedrick [17], RIP is one of the most widely used Interior Gateway Protocols, and is popularly known by the name of a program that implements it *routed* The *routed* software was originally designed at the University of California at Berkeley to provide consistent routing and reachability information among machines on their local networks The RIP protocol is

an implementation of the Bellman-Ford [4] distance-vector routing algorithm within a single autonomous system in the IP-based Internet It is intended to allow hosts and gateways to exchange information for computing routes through an IP network Each participating entity in the RIP runs the *routed* process that sends and receives the RIP routing messages In the RIP, there is a provision to allow *passive* RIP processes A passive RIP member does not send out any messages, however it listens to RIP updates in order to monitor local gateways and to keep its internal routing tables updated All gateways should implement *active* RIP processes (i e , send and receive routing messages) for dynamic routing The Bellman-Ford Routing is briefly described below.

3.1.1 Distance-Vector (Bellman-Ford) routing

The idea behind the vector-distance algorithm to propogate routing information is as follows.

- Each gateway begins with a set of routes for those networks to which it attaches. This set is kept in the form of a table, where each entry identifies a destination network and gives the distance to that network from the gateway, measured in hops
- Periodically, each gateway sends a copy of its routing table to any other gateway, it can directly reach.
- When a route report arrives at gateway K, (see Figure 3 1) from gateway J, K examines the set of destinations reported along with the distance to each. If J knows a shorter route to reach a particular destination, or, if J lists a destination that K does not have in its routing table, or if K currently routes to a destination through J and J's distance to that destnation has changed, K replaces its table entry For example, Figure 3 2 shows an existing table in the gateway K, and an update message from another gateway, J.
- When K updates an entry based on J's route report, for reported distance of N to a destination, K updates the distance field for that destination to N+1 (the distance to reach the destination from J plus the distance to reach J from K).

In the next section a distance-vector style multicast routing algorithm as proposed by Deering [9] is described. For multicast datagram forwarding, more state information about a destination than just the distance is required. This makes the implementation of distance vector based multicast routing algorithms more complex than the unicast implementation

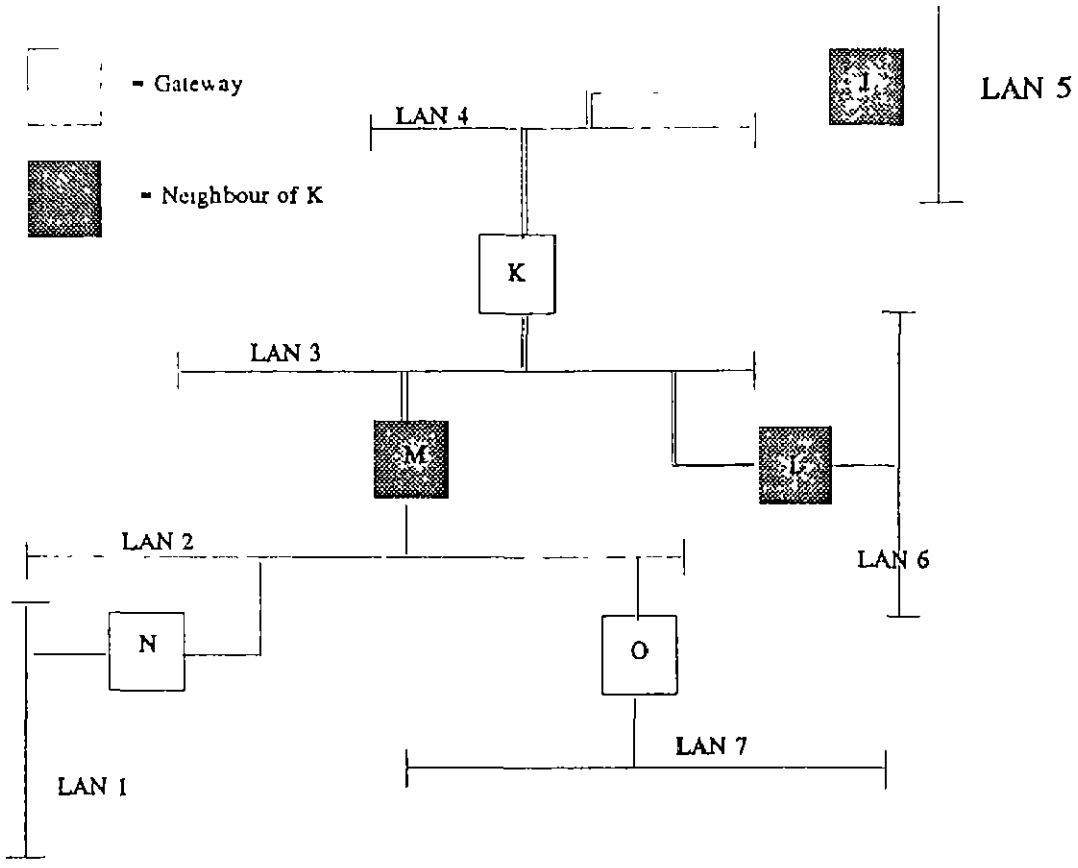


Figure 3 1 Route Reporting in Distance-Vector Algorithm

3.2 Distance-Vector Style Multicast Routing Algorithm

According to Deering, a straight-forward way to support multicasting in a distance-vector routing environment would be to compute a single spanning tree across all of the links and then use some multicast forwarding algorithm to forward multicast datagrams. In a general topology that provides alternate paths, no single spanning tree provides minimum-delay routes from all senders to all sets of multicast groups The multicast routing algorithm proposed by Deering is based on the following observations :

- To achieve the goal of low-delay multicasting, it is required that the multicast packet be delivered along the shortest path tree, from the sender to the members of the multicast group

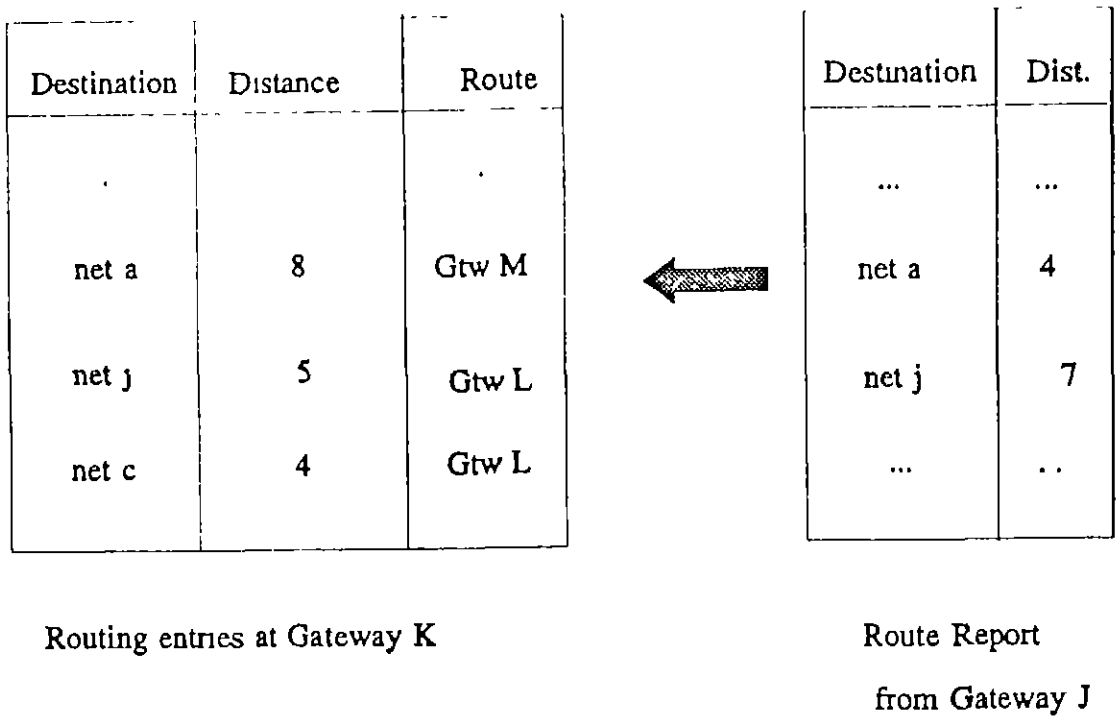


Figure 3 2 Routing Update at Gateway K

- There is potentially a different shortest-path tree from every sender to every multicast group. However, every shortest path multicast tree rooted at a given sender is a subtree of a single shortest path *broadcast tree* rooted at the sender.

Deering has modified the Reverse Path Forwarding algorithm by Dalal and Metcalfe [8], to produce low delay routes from a sender S to members of group G on the network.

3.2.1 Reverse path forwarding

The basic Reverse Path Forwarding algorithm (RPF) is as follows:

> A broadcast packet originating at source S is forwarded by the router, if and only if it arrives via the shortest path from the router back to S. The router forwards the packet to all incident links except the one on which the packet arrived. In networks, where the *length* of each path is the same in both directions (for example when using hop counts to measure lengths), this algorithm results in a shortest path broadcast to all links

As described, reverse path forwarding accomplishes a *broadcast*. To use it for multicasting, a set of internet multicast addresses that can be used as packet destinations, have to be specified Then, the reverse path forwarding can be performed on all packets destinated to such addresses. Hosts choose which group they wish to belong to, and discard all arriving packets addressed to any other group.

3.2.2 Refinements to the RPF algorithm : TRPB

A major drawback of the basic RPF algorithm is that any single broadcast packet may be transmitted more than once across a link subject to the number of routes that share the link. This is due to the forwarding strategy, of flooding a packet out all links other than its arriving link, whether or not all the links are part of the shortest path tree, rooted at the sender Two refinements have been proposed.

- To eliminate the duplicate broadcast packets generated by the RPF algorithm, it is necessary for each router to identify which of its links are child links in the shortest reverse path tree rooted at any given source S. A link is a child to a router X, if X is the next-hop address in the shortest path back to the sender S. Then, when a broadcast packet originating at S arrives via the shortest path back to S, the router forwards it to only the child links for S. To discover the child links, Deering [9] has proposed a technique involving identification of a single *parent* router for each link, relative to each possible source S. The parent is the one with minimum distance to S. This is best understood with the following example illustrated in Figure 3.3.

  Here two routers X and Y are attached to LAN A and router Z is connected to a leaf LAN B. The dotted line represents the shortest path from X and Y to S, with the associated metric Thus router X is 5 hop counts away from S, distance from Y to S is 6 and that from Z to S is also 6

  In the basic RPF algorithm, both X and Y receive a broadcast from S over their shortest path links to S and both of them forward a copy to LAN A. Therefore any hosts attached to A receive two copies of all packets broadcast from S. Router Z will however, forward only one of the copies, the one from X, onto LAN B because X is the next hop-address for S.

  The Parent selection technique works as follows:

  All three routers X, Y and Z periodically send distance-vector routing packets across LAN A, reporting their distance to every destination.

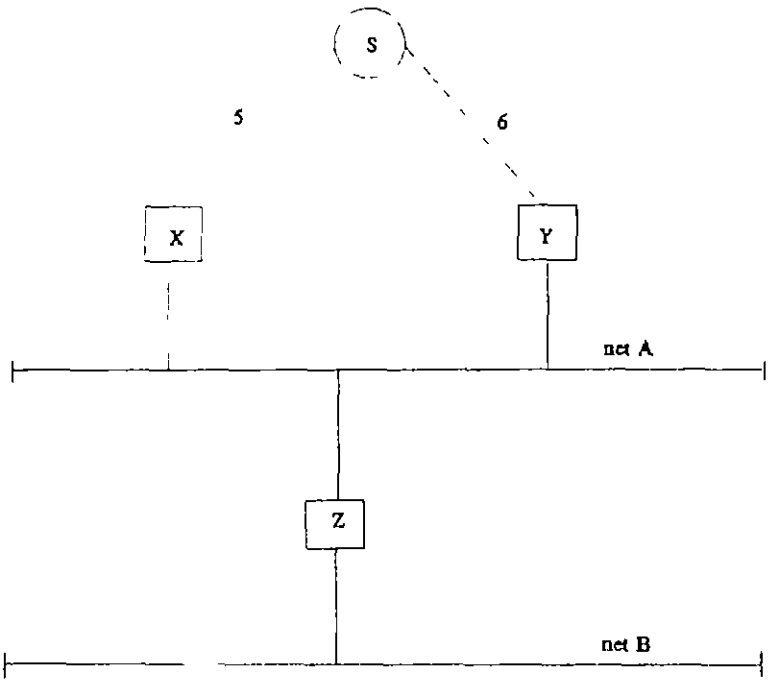


Figure 3 3 TRPB An Example

From these packets, each of them learns that X has the shortest distance to S. Therefore, only X adopts LAN A as a child link, relative to S, and Y no longer forwards superfluous broadcasts from S onto LAN A

This parent-selection technique requires one additional field, `Children` to be added to each routing table entry. `Children` is bit-map with one bit for each incident link. The bit for link l in the entry for destination is set to 1 if l is child link of this router for broadcasts originating at destination.

- In case of extended LANs, it is desirable to conserve network and router sources by sending multicast packets only where they are wanted. This requires that, hosts inform the routers of their group memberships Thus to provide shortest path multicast delivery from source S to members of group G, the shortest-path broadcast tree rooted at S must be pruned back to reach only as far as those links that have members of G.

  For a router to forgo forwarding a multicast packet, over a leaf link that has no group members, the router must be able to

  – identify leaves and

– detect group membership

Leaf links are those child links that no other router uses to reach S.

Referring to Figure 3.3, LAN B is an example of a leaf link for the broadcast tree rooted at S. I the routers periodically send a packet on each of its links, advertising that link is the next-hop to these destinations. Then the parent routers of those links can tell whether or not the links are leaves for each possible destination.

In the earlier example, router Z would periodically send a packet on LAN A saying *this link is my next hop to S* Hence router X, the *parent* of LAN A would learn that LAN A is not a leaf, relative to S In the routing tables, another bit-map field, `leaves` is added to each entry, identifying which of the children links are leaf links. The next step is to identify whether or not the members of a given group exist on those leaves To do this, the hosts periodically report their memberships, using local multicast The routers on the local network keep a list, one for each incident link, of the groups present on that link

The reverse path forwarding algorithm now becomes

```
IF a multicast packet from S to G arrives from
   the next-hop address for S,
     Forward a copy out all child links for S
   except
     The leaf links which have no members of G.
```

In the next section the implementation of the multicast routing protocol as described by Waitzman *et al*[18] is discussed This software is distributed freely, over the Internet, and has been used for the MBONE implementation on the IIT Kanpur LAN.

3.3 DVMRP - Distance Vector Multicast Routing Protocol

Waitzman *et al*[18] have detailed the implementation of a distance vector style routing protocol (DVMRP) for routing multicast datagrams based on Deering's algorithm described above. This protocol is derived from RIP and implements the TRPB algorithm for forwarding multicast packets. DVMRP combines features of RIP with TRPB It is an *Interior Gateway Protocol* suitable for use within an autonomous system, but not between different

autonomous systems It's current implementation does not support routing non-multicast datagrams, so, a router that routes multicast and unicast datagrams, needs to run two routing processes, one for the multicast and the other for the normal unicast and broadcast datagrams The multicast forwarding algorithm requires the buliding of trees based on routing information In addition to the implementation of the TRPB algorithm for datagram forwarding this implementation provides a **tunneling** mechanism to traverse networks that do not support multicasting Some of the implementation issues of the DVMRP are discussed in the following sections

3.3.1 RIP vs DVMRP

The tree building process for datagram forwarding needs more state information that RIP is designed to provide. The most important difference in the two algorithms is at a configuration level. RIP is configured in terms of routing and forwarding datagrams to a particular *destination*. Whereas the DVMRP aims at keeping track of the *return paths* to the *source* of the multicast datagrrams. Thus in the DVMRP context, datagrams are not *forwarded* to the *destination* of the (distance, destination) pair in the routing message, but *originate* from them. This makes DVMRP implementation much more complex as compared to RIP

3.3.2 Format of DVMRP datagrams

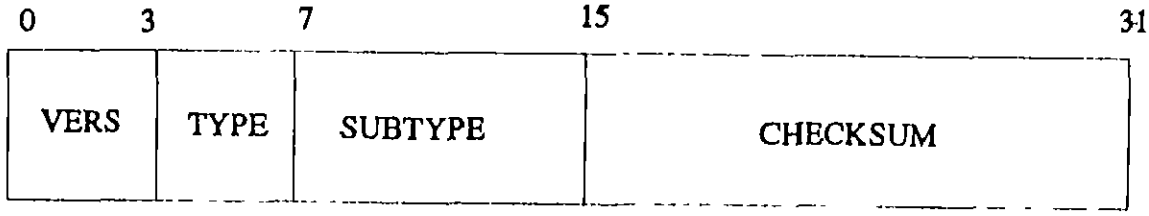


Figure 3.4: Fixed Length IGMP Header of DVMRP Messages

DVMRP uses the Internet Group Management Protocol (IGMP) to exchange routing datagrams. DVMRP datagrams are composed of two portions : a small, fixed length IGMP header and a stream of tagged data The fixed length IGMP header of DVMRP message is as shown in Figure 3 4 The types of DVMRP messages are as follows:

1. Response : provides routes to destinations

2 Request requests routes to destinations

3 Non-membership report(s)

4 Non-membership cancellation

The checksum is 16-bit one's complement of one's complement sum of the entire message, excluding the IP header The element of the stream of tagged data are called *commands* and are multiple of 16 bits The commands are organised as an eight bit command numeric code, with atleast an 8-bit data portion A message that has an error in it will be discarded at the point in prosessing, where the error is detected. The length of the DVMRP message is limited to 512 bytes, excluding the IP header The supported commands are included in the appendix B

3.3.3 The DVMRP tunnel mechanism

DVMRP defines a method for sending datagrams between routes seperated by gateways that do not support multicasting routing A tunnel provides a virtual path between two multicast routers.

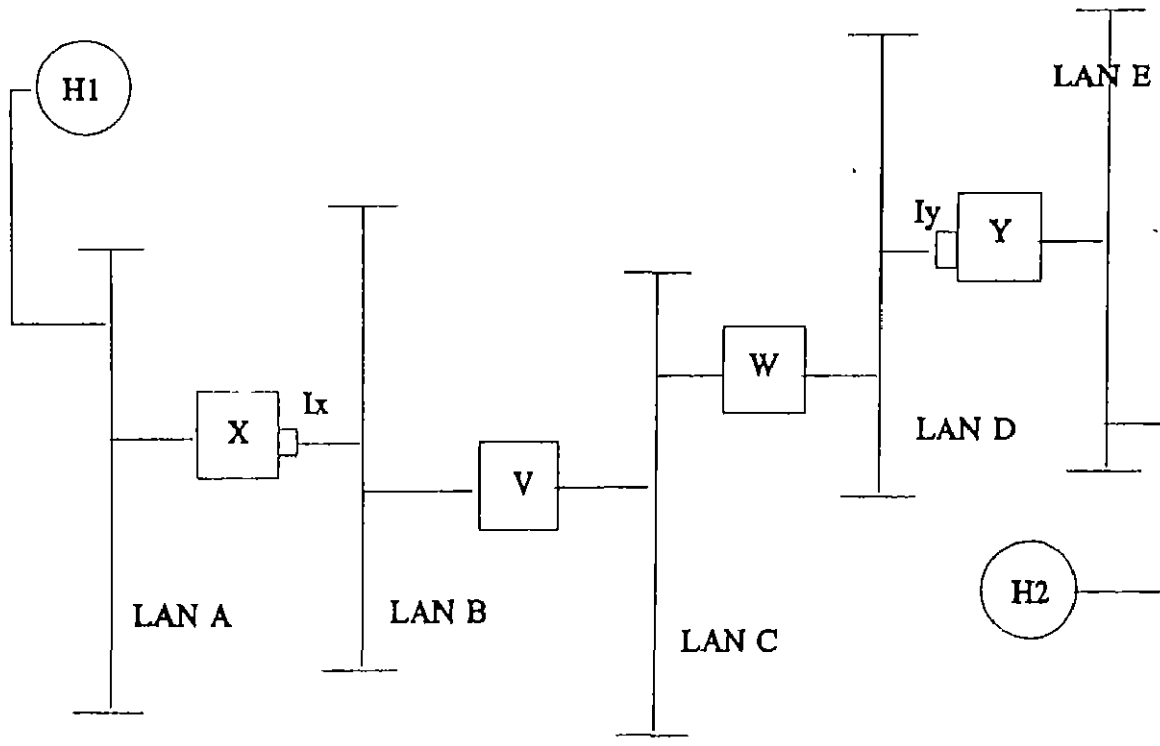


Figure 3.5: Tunnel Configuration - An Example

Tunneling has been considered as transitional mechanism To understand the tunneling mechanism, consider the network in Figure 3.5

Host H1 is on LAN A and Host H2 is on LAN E Routers X and Y are multicast routers and accept multicast IP datagrams Routers V and W are routers which do not support multicast routing, and therefore would discard multicast IP packets if the destination address is a class D internet address To provide the virtual link between X and Y for IP multicast datagrams to flow from X to Y a tunnel is defined between the routers X and Y. A tunnel has the following parameters associated with it ·

- local end point: the local host's internet address
- remote end point. the remote host's internet address
- metric the cost of sending the datagram via the tunnel
- threshold the minimum ttl value that the datagram should have for forwarding this datagram across the tunnel

A tunnel defined in X, from X to Y would have the configuration

- local end point = Internet address of the network interface of X connected to V (Ix).
- remote end point = Internet address of the network interface of Y connected to W (Iy)
- metric = 3
- threshold >=3

A similar tunnel would have to be configured in Y with the local and remote end points defined appropriately.

This tunnel makes the above network configuration look as shown in Figure 3.6. The routers at each end of the tunnel need only agree upon the local and remote end point. The metric and threshold should be so chosen as closely appoximate the no of intermediate gateways between end points. Tunneling is done with a weakly encapsulated normal multicasted datagram The weak encapsulation uses a two element IP Loose Source Route option. Recent implementations do tunneling using **strong** encapsulation, i.e prepending entire new IPheader.

The idea behind the *source route* IP option is that, it provides a mechanism for the sender, to dictate a path through the Internet. In the *loose source route* form, a sequence of IP addresses are included which specify the route Multiple network hops are allowed between successive address on the

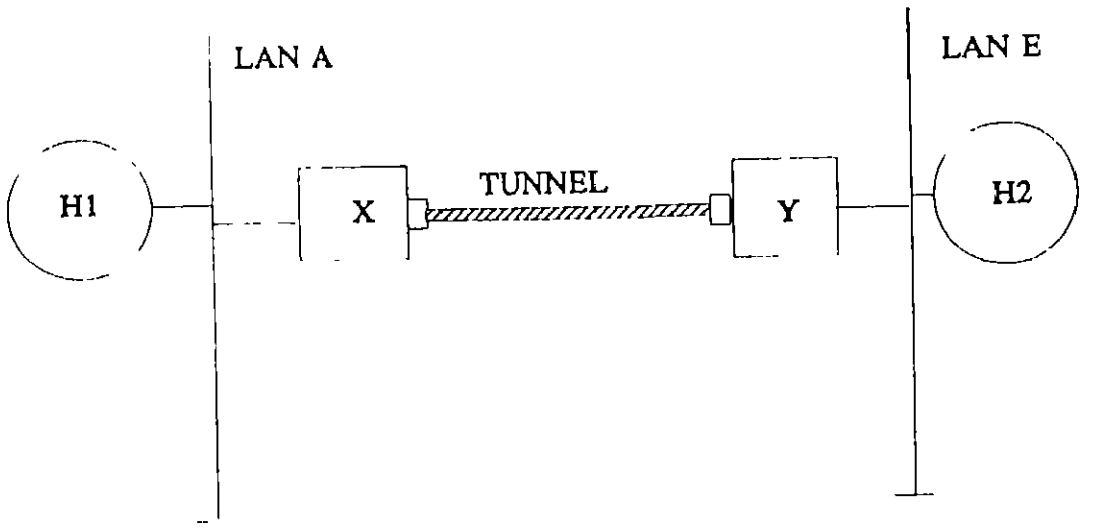


Figure 3.6: A Virtual Network Configuration

list. The format of the IP LSR is shown in Figure 3.7. The code field contains the option no. and class. The length field specifies the total option length including the first 3 octets. The pointer specifies the offset within the option, pointing to the first address. Routing messages are carried through tunnels, but a route is not created for the tunnel. The routing messages are sent as unicast datagrams directly to the remote end-point without using the IP LSR option.

For the tunnel mechanism to work properly, the intermediate gateways should ignore the LSR option and the remote-end point should detect the LSR option. The former is possible because, when an intermediate gateways receives a datagram, it examines the destination address. This address will not match its own address, and therefore, the receiving gateway will not examine the LSR option, but pass on this packet to the next gateway in the path to the destination. When the remote end point receives the datagram, the datagram destination address matches its own address. It then looks at the next LSRR option address, since the source route has not exhausted. That address is a multicast address. Because the hosts are not allowed to put multicast address in source routes, the gateway can infer that the LSRR is for tunneling. In the following section some implementation requirements of the routing algorithm for routing of multicast packets are studied.

3.3.4 Virtual Interfaces and Virtual Networks

While the DVMRP can express routes to hosts, the forwarding and routing algorithm only support network and subnetwork routing. To incorporate the

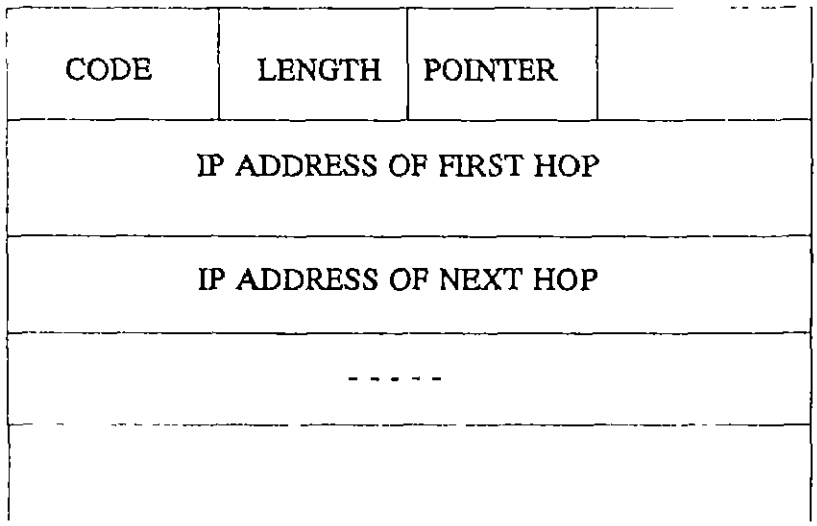


Figure 3.7: IP LSRR Option

tunnels as valid network interfaces, the concept of virtual interfaces and virtual networks has been introduced. A physical interface, such as an Ethernet card or a tunnel local-end point, is termed as a **virtual interface** in this specification. A route to destination is through a virtual interface. A **Virtual Network** refers to a physical network or tunnel, however, routes only reference physical networks The TRPB algorithm forwards multicast datagrams by computing the shortest path tree from the source (a physical network) to all possible destinations (member of a group). Each multicast router determines its place in the tree, relative to particular source, and also determines which of its virtual interfaces are in the shortest path tree. The datagram is forwarded to all *child virtual interfaces*, except those *leaf interfaces* which have no members of the group. The building of the tree, discovering membership and leaf interfaces is done by exchange of routing messages with neighbouring routes The state is maintained in a routing table containing one entry, for each reachable destination, over the virtual interface

3.3.5 Determining leaf interfaces

If any neighboring router, considers a given virtual network in the path to a given destination, then the virtual network is not a leaf. Otherwise, it is a

leaf This is a voting function If a route, with a metric poisoned by split horizon processing [17], is sent by some router, then that router uses that virtual network as the uptree path for that route (i e , that router votes that the virtual network is not a leaf relative to the route's destination) Since the number of routers on a virtual network is dynamic, and since all received routing updates are not kept by routers, some heuristic mechanism is needed to determine when a network is a leaf. DVMRP samples the routing updates on a virtual interface while a hold down timer is running, which is for a time period of LEAF_TIMEOUT seconds. There is one hold-down timer per virtual interface If a route is received with a metric poisoned by split horizon processing [17] while the hold-down timer is running, or at any other time, then the appropriate virtual interface for that route is *spoiled*– it is not a leaf For every route, any virtual interface that was not spoiled by the time the hold-down timer expires is considered a leaf.

3.3.6 Routing table entry

A route entry should have the following in it·

1. Destination address (a source of multicast datagrams) *
2. Subnet mask of the destination address *
3. Next-hop router to the destination address
4. Virtual interface to the next-hop router *
5. List of child virtual interfaces *
6. List of leaf virtual interfaces *
7. A dominant router address for each virtual interface
8. A subordinate router address for each virtual interface
9. Timer
10. Set of flags that indicate the state of the entry
11. Metric
12. Infinity

The lines that are marked with * indicate fields that are directly used by the forwarding algorithm. The lists of child and leaf interfaces can be implemented as bitmaps.

3.3.7 Thresholds

One concept in Internet Multicasting is to use *thresholds* to restrict which multicast datagrams exit a network. Multicast routers on the edge of a subnetted network or autonomous system may require a datagram to have large TTL to exit a network. This mechanism keeps most multicast datagrams within the network, reducing external traffic. An application that wants to multicast outside of its network, would need to give its multicast datagrams at least a TTL of the sum of the threshold and the distance to the edge of the network (assuming TTL is used as a hop count within the network). A configuration option, should allow specifying the threshold, for both physical interfaces and tunnels

3.3.8 Sending routing messages

DVMRP routing messages can be used for three basic purposes:

- to periodically supply all routing information,
- to gratuitously supply routing information for recently changed routes or
- supply some or all routes in response to a request.

Routing messages sent to physical interfaces should have an IP TTL of 1. A number of timeouts and rates are used by the routing and forwarding algorithms. The values for these are given in appendix B The rules for when to send routing messages are given in chapter 4.

3.3.9 Receiving routing messages

A router should know the virtual interface that a routing message arrived on. Because the routing message may be addressed to the all-multicast-routers IP address, and because of tunnels, the incoming interface cannot be identified merely by examining the message's IP destination address The route report processing is discussed in detail in Appendix B.

3.3.10 Discovering neighbours

Neighbours are discovered at startup during initialization when neighbour probe packets are sent over all multicast capable interfaces of the router. Tunnel remote-end points are also neighbours. A list of the neighboring multicast routers on every attached network should be kept. The information can be derived by the DVMRP routing messages that are received A neighbor that has

not been heard from in NEIGHBOR_TIMEOUT seconds should be considered to be down

3.3.11 Local group memberships

A multicast router must keep track of group memberships on the multicast-capable networks attached to it, as required in the IGMP implementation. Every QUERY_RATE seconds an IGMP membership request should be sent to the **All Hosts multicast address (224.0.0.1)** on each network by a designated router on that network. The IGMP membership request will cause hosts to respond with IGMP membership reports after a small delay Hosts will send the report for a group to the group's multicast address The membership requests should have an IP TTL of 1

The routers on a network elect a single router to send the queries. The designated router is the router with the lowest IP address on that network. Upon startup a router considers itself to be the designated router until it learns (through routing messages) of a router with a lower address To learn about the group members present on a network at startup, a router should multicast a number of membership requests, seperated by a small delay The authors suggest sending three requests seperated by four seconds.

The multicast router must receive all datagrams sent to all multicast addresses. Upon receiving an IGMP membership report for a group from an interface, it must either record the existence of that group on the interface and record the time, or update the time if the group is already recorded The recorded group memberships must be timed-out. If a group member report is not received for a recorded group after MEMBERSHIP_TIMEOUT seconds, the recorded group should be deleted.

Several issues such as the time-out values and threshold depend on the network span and bandwidth

In the next chapter, the Multicast Backbone implementation using the IP multicast, IGMP and DVMRP, is discussed.

Chapter 4

IP Multicast Backbone Implementation Details

In this chapter the Internet Engineering Task Force (IETF) Multicast Backbone (MBONE) implementation is first studied. Next, the IP multicast backbone implementation on the Campus network is presented Some of the LAN enhancements for group communications have been demonstrated using public domain software available over the Internet The concluding part of this chapter gives the details of this software used for demonstration purpose

4.1 The IETF Multicast Backbone Configuration

The MBONE was designed and configured by the IETF to support multicast transmission of IP datagrams carrying live *audio* and *video* from the IETF meeting sites to remote destinations over the world. It has been envisaged as an IP-multicast testbed to support continued experimentation between various Internet sites. The MBONE experiment began as a co-operative effort between a few Internet sites and later became very popular among the Internet community. Being an experimental implementation there is very little documentation and most of the issues related to the MBONE have been explained in MBONE - Most Frequently Asked Questions, (FAQ)[6]

4.1.1 MBONE - A virtual network

The MBONE is a virtual network layered on top of parts of the existing physical Internet, configured to support the routing of multicast packets to remote destinations on the Internet. The network is virtual in the sense that, there is no seperate physical network to carry the multicast packets Portions

of the existing network have been configured for multicasting without any physical changes to the network

The MBONE consists of *islands* or *networks* which support multicast, connected by virtual links called *tunnels* (see Chapter 2 for the tunnel definitions) The tunnel end-points are typically workstation class machines, having operating system support for IP multicast, and running the Distance Vector Multicast Routing DVMRP daemon process *mrouted*

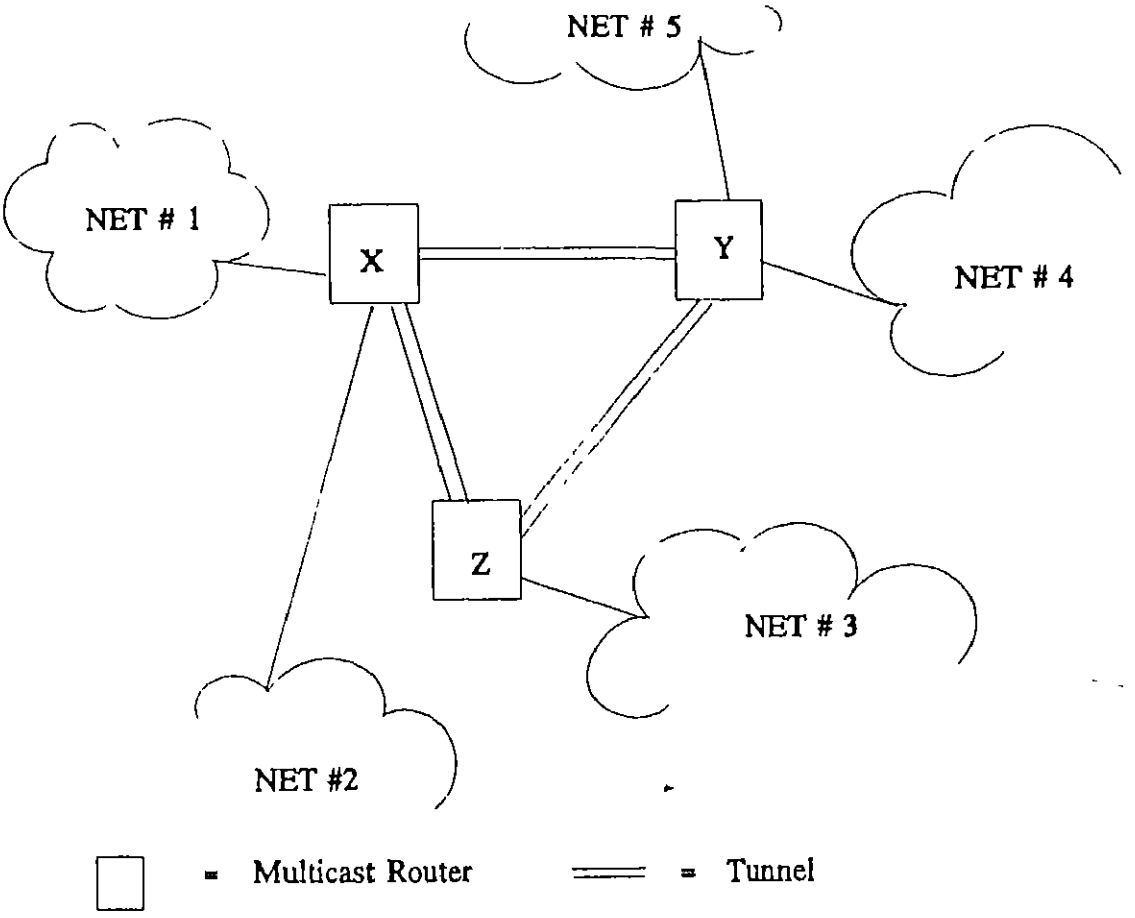


Figure 4 1: A Simple MBONE Configuration

A simplified MBONE configuration is as shown in Figure 4.1. The tunnel mechanism is implemented by either using an IP LSRR option or by IP encapsulation. The IP encapsulataion method adds an extra IP header with the source and destination address of the tunnel end-points to every multicast IP datagram The IP LSRR option has alr dy been described in Chapter 3. Due to the wide geographic span of MBONE, the designers have given a careful thought in planning the MBONE topology across the world. An effort

has been made to minimise the traffic between continents The detailed description with the hardware and software requirements to join the MBONE, are given in the [6] A few of the interesting issues involved in the MBONE configuration are discussed below

4.1.2 Topology of MBONE

Within a continent, the topology of MBONE is a combination of **mesh** and **star** The regional and backbone networks are linked by a mesh of tunnels to form the topmost level of the MBONE heirarchy The tunnels are configured among machines executing the multicast routing daemon *mrouted*. These machines are primarily located at the interconnection points of the backbone and the regional networks. For robustness, redundant tunnels with higher metric are also configured.

Each of the regional network has a **star** hierarchy hanging off its node of the mesh, to fan out and connect customer networks. The setup of the MBONE toplogy has been through continued co-operative efforts among the participants . At the topmost level is the configuration of the **mesh** which is managed by the IETF through four regional centres These four regions are identified on a geographical basis viz

1 North America

2. Europe

3. Australia

4. Others

The IETF has set up seperate mailing lists for each of these regions to co-ordinate customer network connections to the MBONE

Another important issue in the MBONE design is the bandwidth considerations, and support for real-time traffic. The MBONE design considerations for these are given below.

4.1.3 Traffic levels on the MBONE

The primary aim of the MBONE set-up was to carry live audio and video. With a data rate of 64kb/s PCM encoded audio and 64 kb/s - 128 kb/s H.261 [5] compatible video, a design bandwidth of 500 Kbps was chosen. During the IETF transmissions a traffic level of 100-300 Kbps was anticipated. At the peak level, 5 simultaneous conversations were expected. Therefore, the design bandwidth of 500 Kbps was justified.

Since each tunnel carries a seperate copy of every packet, the design bandwidth needs to be multiplied by the number of tunnels passing over any given link For this reason, each multicast router, is configured to connect a maximum of 10 links, and the topology of MBONE is so designed as to configure at most two tunnels over any intercontinental link It is expected that most of the MBONE nodes connect with lines of atleast T1 (1.544Mbps) speed.

To restrict the traffic over slower speed links, each tunnel has an associated threshold against which the IP packet's *time to live* (TTL) field is compared. Higher bandwidth sources such as video transmit with smaller ttl so that they can be blocked from transmitting over lower speed links, whereas lower bandwidth sources such as compressed audio are allowed to pass through

The *threshold* is the minimum TTL value required for a multicast datagram to be forwarded to the given interface or tunnel. For example, video applications generate IP packets with a TTL of 127. However, if the tunnel is configured with a *threshold* greater than 127, say 128, these packets will be dropped

At present the mesh configuration is completely handled through co-operative efforts on the part of network providers within each region as well as at higher levels of the MBONE hierarchy. Until a mechanism of MBONE configuration using some network management protocols is available, the network providers and the customer networks have to interact to provide the MBONE connections

In this project, we have emulated MBONE essentially to study the various issues involved in providing group communications services over the Campus network. The Campus network is an extended LAN-topolgy Three host machines on three different subnets have been configured to run the *mrouted*. In the next section, the IIT Kanpur LAN configuration is studied.

4.2 IIT Kanpur Campus LAN Architecture

The IITK LAN consists of several Ethernet segments connected to a thick ethernet backbone The LAN configuration is as shown in the Figure 4.2.

The segments are interconnected by a mix of IP routers, bridges (both developed at IIT Kanpur) and repeaters. The LAN covers the entire academic area. A few sites are also connected through the digital multikey telephone (DMKT)-based serial data links. This LAN is connected through a CISCO router to the ERNET WAN via two serial ports and modems to DOE Delhi and IIT Kharagpur. Outside e-mail users are connected via dialup modems to a gateway system. The IIT Kanpur Campus network site is a part of the Educational and Research Network (ERNET). The ERNET has been allocated a single IP class B network number (144.16). Each ERNET site has been alloted 32 class B subnet numbers. The IITK uses IP subnets rang-

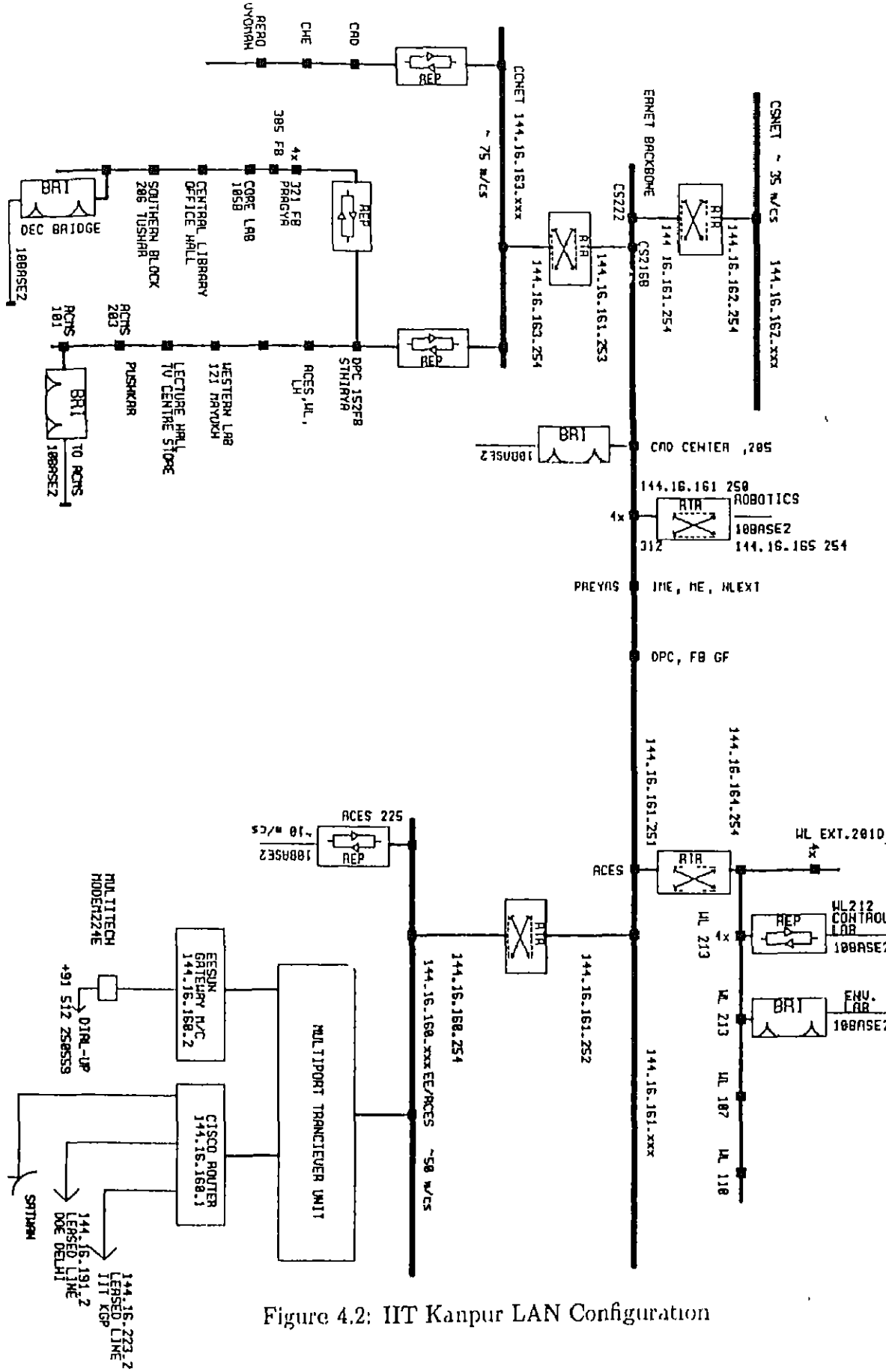


Figure 4.2: IIT Kanpur LAN Configuration

ing from 144 16 160 xxx to 144 16 191 xxx Presently, 4 class B IP subnets are being used in the campus network These are the ERNET backbone (144.16 161 xxx), CSENET (144 16.162 xxx), CCNET (144 16 163.xxx) and ACESNET/EENET (144.16 160 xxx) The ERNET backbone connects the other three subnets through IP routers It is planned to upgrade the routers to support class C level filtering so that overall management of the address space is simplified.

With the above configuration of the Campus network, given in Figure 4 3 is the desired MBONE virtual network on the campus LAN.

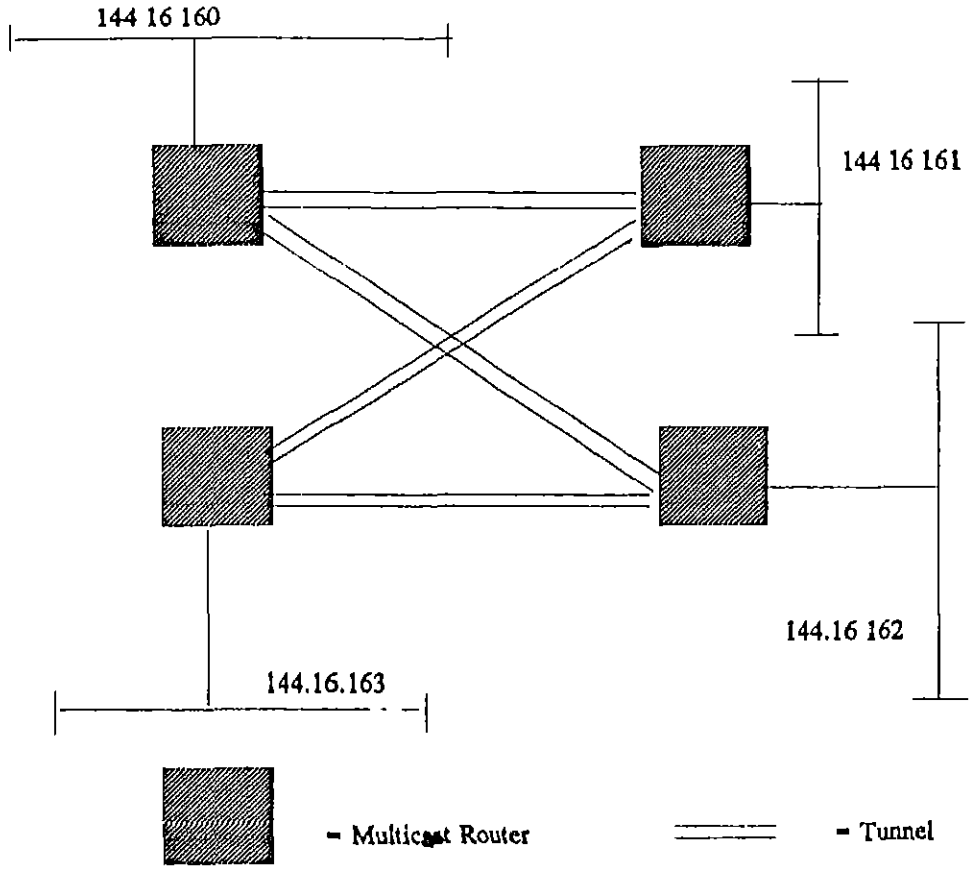


Figure 4.3: Desired IITK-MBONE Network

For the trial implementation three machines have been chosen on three different subnets and linked by tunnels. The details of the trial configuration are given in the following section.

4.3 Configuration of MBONE on Campus LAN

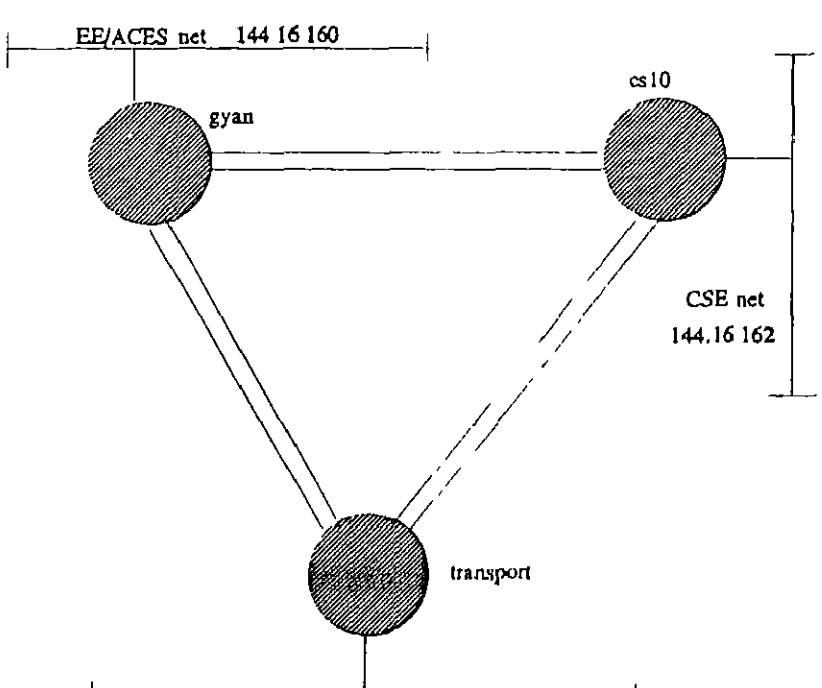


Figure 4 4. IITK-MBONE Experimental Setup

This configuration provides multicast IP datagram transmission and routing support for hosts on the CSENET, EENET and ERNET backbone. The IP multicast routers are configured as shown in Figure 4 4, and the corresponding physical interconnection is as shown in Figure 4.5. *gyan* is a SunSPARC10 machine with sun4m architecture and SunOS 4 1.3 operating system, on the subnet 144 16.160. This machine acts as a multicast router for the ACES/EENET subnet. The host *transport* is a machine on the subnet 144.16 161 and is also a SunSPARC10 with same configuration as *gyan* It routes all the multicast traffic to/from the ERNET backbone. The third machine *cs10* is a SunSPARC1+ with sun3 architecture and sunOS 4.0 operating system. This machine is configured to route the multicast traffic to/from the CSENET subnet The features of the Campus-MBONE implementation are given in detail

below

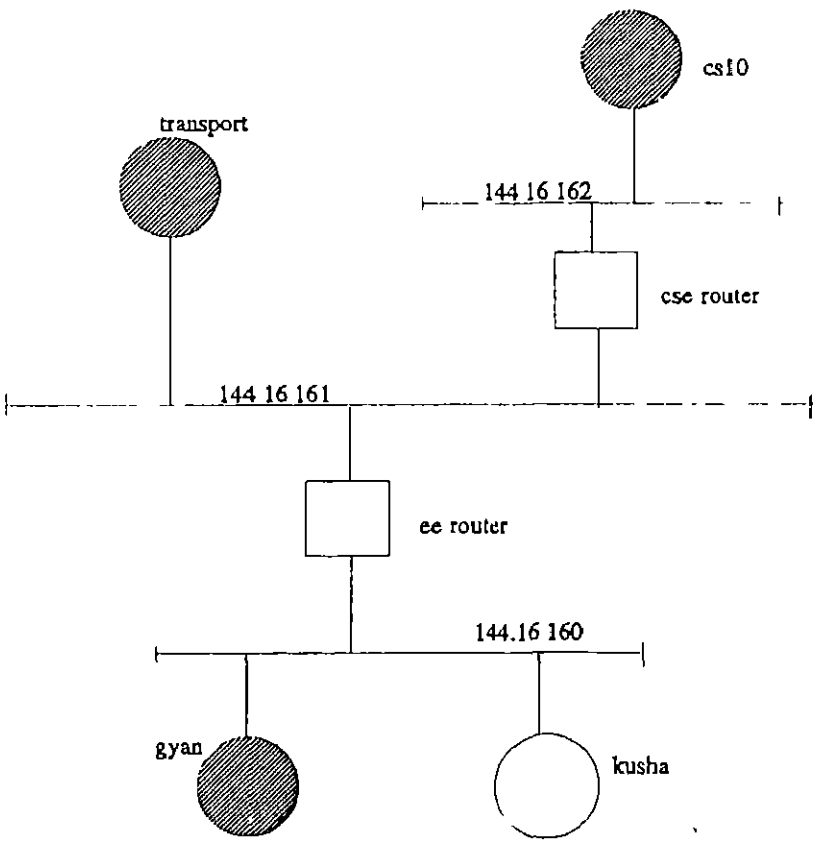


Figure 4 5 IITK-MBONE The Corresponding Physical Interconnections

4.3.1 Features of the IITK-MBONE

The characteristic features of the IITK-MBONE are as follows·

1 No modification made to the existing IP routers in the LAN. These route the unicast packets as before. The existing IP routers implement passive RIP and are being upgraded to run the active RIP processes These routers allow loose source routing of the IP datagrams. The multicast routers send the multicast datagrams using the IP LSRR or by IP encapsulation to the remote end-points

2. Typically, a router has atleast two physical network interfaces. In addition, for multicast routing, the concept of a virtual interface (see section

3 3 4) allows a router to build one or more tunnels over the same physical interface These tunnels act as links to different subnets and therefore for non-trivial multicast routing the number of virtual network interfaces and the physical network interfaces must be atleast two To allow hosts with a single physical interface to perform the routing function, the *multicast routing daemon* has been modified to build two virtual interfaces over the same physical interface.

3. Host machines *cs10*, *gyan* and *transport* and *kusha* have been configured to also support host IP multicast, conforming to the level 2 IP multicast as specified in the standard [19] for IP multicast These machines can send and receive IP multicast datagrams

4. The IP multicast hosts and routers run the Internet Group Management Protocol (IGMP) The routing updates and membership reports are sent using the IGMP message formats as specified in [19]

5. The public domain software sd for multicast session creation and management has been ported on all IP multicast hosts, to demonstrate some features of the MBONE

4.4 The Implementation Details

The implementation of the IP multicast and multicast routing, has been done in two stages

- Host extensions for IP multicast
- Multicast routing daemon *mrouted* implementation

The IP multicast host extensions were available from anonymous ftp server on the host `gregorio.stanford.edu` in the directory `vmtp-ip` This software when compiled, modifies the kernel to support IP multicast To understand the IP multicast implementation, in the UNIX domain, a brief summary of the internal structure of the network subsystem provided by the UNIX environment is included in Appendix C. More details of the Unix Networking Environment can be found in [13] or [24].

The modifications made to the network interface module, and the communication modules of the network subsytem in each of the hosts on which IP multicast capability has been added are explained below.

4.4.1 The network interface

The state of the network interface and certain externally visible characteristics are stored in the Interface Flags field as shown in Table 4.1. These flags are set by the system with `ioctl` [13] requests. The modifications made in the network interface for multicasting are as follows:

- The flag `IFF_MULTICAST` is set using the `attach` routine at boot time.
- The driver is configured to handle changes in the `IFF_PROMISC` and `IFF_MULTI` interface flags. These are needed when co-ordinating the address filter list of the interface.
- Two new ioctls `SIOCADDMULTI` and `SIOCDELMULTI` are added, to add or delete link-level (eg. Ethernet) multicast addresses accepted by a particular interface. The address to be added or deleted is passed as a `sockaddr` structure of family AF_UNSPEC within the standard `ifreq` structure. These `ioctls` can be used by protocols other than IP and require superuser privileges.
- For Ethernets, the function `ether_addmulti()` and `ether_delmulti()` are added, which can be used to maintain a list of multicast addresses. Macros `ETHER_FIRST_MULTI()` and `ETHER_NEXT_MULTI()` are added, to scan the list when updating the address filter.

NETWORK INTERFACE FLAGS

FLAGS	Description
IFF_UP	Interface is available for use
IFF_BROADCAST	broadcast is supported
IFF_DEBUG	enable debugging in the interface software
IFF_LOOPBACK	this is a software loopback interface
IFF_NOTRAILERS	interface should not use trailer encaps
IFF_RUNNING	interface resources are allocated
IFF_NOARP	interface should not use ARP protoc
IFF_POINTTOPOINT	interface is a point-to-point link

These modifications are independent of the network layer protocol The TCP/IP communication protocol specific modifications can be best understood on the basis of the model of the host IP implementation as shown in Figure 2 4

4.4.2 Extensions to the IP service interface

The interface from the socket routines to the IP is through user requests and control output routines defined in the IP protocol switch table This implementation supports IP multicast on raw sockets and datagram sockets of the AF_INET family The raw socket allows privileged users direct access to the IP protocol.
The details of UDP socket SOCK_DGRAM interface are given in [24] The IP service interface has been modified to provide three new socket options, for specifying parameters in the IP header of the multicast datagram. These are described below ·

- `IP_MULTICAST_TTL` .

  This option has been added to limit the *scope* of the multicast packet to a span decided by the `ttl` field of the datagram header This limits unnecessary transmissions to networks beyond the scope of the multicast group Using this option the `ttl` field in the IP header can be set from 0 to 255, in order to control the scope of the multicast

```
u_char ttl;
setsockopt(sock, IPPROTO_IP, IP_MULTICAST_TTL,
                  &ttl, sizeof(ttl));
```

  Multicast datagrams with a `ttl` of 0 are not transmitted on any subnet, but only delivered locally

- `IP_MULTICAST_IF`

  Each multicast transmission is sent from a single network interface set to default by the network administrator This socket option allows overriding the default multicast interface for subsequent transmissions from a given socket

```
struct in_addr addr;
setsockopt(sock, IPPROTO_IP, IP_MULTICAST_IF,
                  addr, sizeof(addr));
```

where addr is the local IP address of the desired outgoing interface. To revert to the default, the addr INADDR_ANY should be used with the same socket option. The local IP address of a particular interface can be obtained using the SIOCGIFCONF ioctl. To determine if an interface supports multicasting, the interface flags can be fetched via the SIOCGIFFLAGS ioctl to check the IFF_MULTICAST flag.

- IP_MULTICAST_LOOP.

  If a multicast datagram is sent to a group to which the sending host itself belongs (on that outgoing interface), a copy of the datagram is, by default looped back by the IP layer for local delivery. To give the sender explicit control over whether subsequent datagrams are looped back.

```
u_char loop;
 setsockopt(sock, IPPROTO_IP, IP_MULTICAST_LOOP,
                  loop, sizeof(loop));
```

  where loop =1 to enable loopback, loop =0 to disable loopback.

  This option provides a performance benefit for applications that may have, no more than one instance on a single (such as a router or a mail daemon), by eliminating the overhead of receiving their own transmissions. It should not be used by applications for which there may be more than one instance on a single host (such as a conferencing program) or for which the sender does not belong to the destination group (such as a time querying program).

- IP_ADD_MEMBERSHIP

  Before a host can receive IP multicast datagrams, it must become a member of one or more IP multicast groups. A process can ask the host to join a multicast group using this socket option :

```
struct ip_mreq mreq;
setsockopt(sock, IPPROTO_IP, IP_ADD_MEMBERSHIP,
                 mreq, sizeof(mreq));
```

  where mreq is the following structure :

```
struct ip_mreq {
  struct in_addr imr_multiaddr;
  struct in_addr imr_interface;
 }
```

Every membership is associated with a single interface and it is possible to join the same group on more than one interface To choose the default interface the imr_interface should be INADDR_ANY, to choose a particular interface the imr_interface should be the host's local address for that interface. Upto IP_MAX_MEMBERSHIPS (currently 20) memberships may be added on a single socket. The IP service interface invokes this option on the

```
JoinHostGroup(group_address, interface)
```

call from the upper layer

- IP_DROP_MEMBERSHIP

```
struct ipmreq mreq;
setsockopt(sock, IPPROTO_IP, IP_DROP_MEMBERSHIP,
                    mreq, sizeof(mreq));
```

where mreq is the same structure as defined above The membership associated with a socket is also dropped when the socket is closed or the process holding the socket, is killed. More than one socket may claim membership to a group, and therfore the host remains a member of the group until the last membership is dropped This option is invoked by

```
LeaveHostGroup(group_address, interface)
```

call from the upper layers. The membership associated with a socket does not necessarily determine which datagrams are received on that socket Incoming multicast datagrams are accepted by the kernel IP layer if any socket has claimed membership in the destination group of the IP datagram Delivery of the multicast datagram to a particular socket is based on the destination port (or protocol type, for raw sockets), just as with the unicast datagrams To receive multicast datagrams sent to a particular port, it is necessary to bind to the local port, leaving the local address unspecified (i e. IN_ADDR_ANY).

- SO_REUSEADDR

More than one process may bind to the same SOCK_DGRAM UDP port if the bind is preceded by

```
        int one = 1;
setsockopt(sock, SOL_SOCKET, SO_REUSEADDR,
                    &one,sizeof(one));
```

In this case, every incoming multicast or broadcast UDP datagram destined to the shared port will be delivered to all the sockets bound to that port. The `SOCK_RAW` does not need this option since no `bind()` is required in the raw sockets.

4.4.3 Extensions to the IP module

To support multicasting, the IP module must be extended to recognize IP host group addresses when routing outgoing datgrams.

This implementation includes the following logic

```
if  IP_destination is on the same local network
or  IP_destination is a host group
    send datagram locally to  IP_destination
else
    send datagram locally to GatewayTo(IP_destination)
```

If the sending host is itself a member of the destination group, a copy of the outgoing datagram must be looped back for local delivery unless inhibited by the sender.

A check is made when sending IP datagrams, that a group address is not placed in the source address field on anywhere in the source routing option of the outgoing datagram.

To support the reception of the multicast IP datagram, the IP module is extended to maintain a list of the host group memberships associated with each network interface.

Associated with each group is a *count* of the number of members of that group. The list of host group memberships associated with a network interface is updated in reponse to the `JoinHostGroup()` or `LeaveHostGroup()` requests from upperlayer protocols. On the first request to join, and the last request to leave a group, the local network module is notified, so that it may update its multicast filter. In addition to the above changes, hosts with level 2 conformance to the IP multicast standard [19] implement the host-IGMP as explained in Appendix A, for informing the local multicast routers of its group memberships.

4.4.4 Extensions to the network service interface

As explained in the Section 2 3 1 of Chapter 2, the mapping of the IP host group address to the Ethernet multicast address is done by the Network service interface Incoming local network multicast packets are delivered to the IP module using the same `Receive_Local` operation as the unicast packets. To allow the IP module to tell the local network module, which multicast packets to accept, the local network service interface is extended to provide two new operations

```
JoinLocalGroup(group_address)
LeaveLocalGroup(group_address)
```

where the `group_address` is an IP host group address. These invoke ioctl calls `SIOCADDMULTI` and `SIOCDELMULTI` at the link level as discussed in the Section 4 4.2

To summarize, the extensions to the host IP implementation provide the following new options to the socket layer

- add membership to a group
- delete membership from a group
- specify a ttl value in the multicast Ip datagram header
- specify loopback/no loopback of the IP datagram
- specify the interface on which to send/receive the multicast IP datagrams

This implementation also incorporates the Host IGMP implementation.

In the next section, we outline the multicast routing daemon implementation along with the modifications made to tailor the code for the IITKanpur LAN environment

4.5 The Multicast Routing Daemon - *mrouted*

Mrouted is an implementation of the Distance Vector Multicast Routing Protocol explained in the Chapter 3, as per [18] This daemon process maintains the topological knowledge via a distance vector routing protocol, upon which it implements a multicast forwarding algorithm called the Truncated Reverse Path Broadcast (TRPB), see section 3.2.2

To route the multicast packets over the campus LAN and yet leave the existing (unicast) routers untouched, required either

- IGMP_HOST_MEMBERSHIP_REPORT : In response to a IGMP_HOST_MEMBERSHIP QUERY, every host which has atleast one group, responds with a report message as per the state diagram of IGMP (see Appendix A) On receipt of the report, the DVMRP router updates the group memberships and records or updates the time After MEMBERSHIP_TIMEOUT seconds, the recorded group is deleted if not heard of

The various timer values are given in the Appendix B The algorithms for the route report processing and the algorithm for manipulating the leaf and child links is included in the same

4.5.2 Multicast packet forwarding algorithm

The forwarding algorithm is as follows

```
   IF  (IP TTL < 2)
   THEN  CONTINUE with the next datagram,
   find the route to the source of the datagram,

   IF (no route exists)
   THEN  CONTINUE with the next datagram;

   IF ( the datagram was not received on the next-hop
            virtual-interface for the route ) ·
   THEN  CONTINUE with the next datagram;

   IF ( the datagram is tunneled )
   THEN  reconstruct the correct source and destination
                IP addresses and adjust the
                IP header length;

   IF ( the datagram is destined to group 224.0.0.1 or
  224.0.0.4 )
   THEN  CONTINUE with the next datagram;

FOR each virtual interface V

   DO  IF ( V is in the child list for the source of the
      datagram )
       THEN IF (V is not in the leaf list of the source)
            OR (there are members of the destination
                group on V )
```

1 dedicated machines with atleast two physical interfaces to execute the mrouted code or

2 modification in the mrouted code to allow multicast hosts with a single physical interface to perform the routing function by building two virtual interfaces over a single physical interface

The second alternative was chosen and has been implemented on *gyan*, *transport* and *cs10*.

The unicast and multicast routing is done by two different processes to route multicast packets to *subnets*, modifications were made to the `structure uvif_array` (this stores the complete details of every configured Virtual interface) in the *mrouted* to allow subnetting. The changes have been done in the tt mrouted/config.c, mrouted/vif c, mrouted/route.c and the related header files.

The next section gives a brief explanation of the DVMRP router implementation. The details are given in [6]. Few of the implementation details are included in the Appendix B.

4.5.1 DVMRP implementation

The route propagation is done through the DVMRP route reports Following DVMRP message types are used to build the routes.

- DVMRP_PROBE : After initialization *mrouted* sends query packets to its valid neighbours for checking their status. If the neighbour is up and reachable, it sends a route update to the initiating router If a neighbour is not heard from in NEIGHBOUR_TIMEOUT seconds, it is considered down by the *mrouted*

- DVMRP_REPORT : A route report with all its routing information is sent to all its virtual interfaces

  1. Every FULL_UPDATE_RATE seconds .
  2 Every TRIGGERED_UPDATE_RATE seconds, whenever a route changes.
  3. When a DVMRP router is restarted
  4. When the DVMRP router is about to terminate execution.

The types of IGMP messages are.

- IGMP_HOST_MEMBERSHIP_QUERY . Every QUERY_RATE seconds an IGMP membership request is sent to the *all_hosts_group* address if the router is designated as querier on that network.

```
THEN IF ( IP TTL > V's threshold )
        THEN subtract 1 from the IP TTL and
  forward the datagram out V;
```

The routing table entries have already been mentioned in Chapter 3

4.6 Demonstration Set-up

The program SESSION DIRECTORY sd which announces, creates and manages multicast sessions has been used A brief summary of sd follows

4.6.1 Session Directory -sd

Sd provides·

- A dynamically updated list of available sessions (e g., vat audio conferences, nv or ivs video conferences, or wb whiteboard conferences). A brief summary of these is given in the next section.
- A way to join any available session(s)
- A way to create and advertise new session(s).

Executing sd creates an X window sd multicasts its own advertisement to be picked up by anyone else running sd. At the same time sd also picks up information about the current sessions on the network and displays them for for the user to participate if desired The audio, video and shared X window (Whiteboard) programs can be invoked with the right parameters through sd The user can open a new session or become a member of the already setup sessions sd starts up the appropriate tools for the session (vat, nv, ivs and/or wb), that have been specified.

When sd creates new sessions, it chooses by default a unique, nonconflicting multicast address, port and conference ID for the session. The user can choose his/her own address, port or conference ID, to setup new sessions also. The address can be either an IP address or DNS name. Port and ID must be integers in the range 1 to 65535.

By default, only audio media is selected for the session. If you want to announce that the conference contains video and/or a whiteboard, the choice has to be explicitly specified

While creating sessions scope lets a user set the multicast ttl of the session

Given below is a brief summary of the voice packetizer vat, and the X11 based tool for shared drawing surface, wb (Whiteboard), which have been used for the demonstration

- Voice Audio Tool - `vat`

  `Vat` is an X11-based audio teleconferencing tool which allows users to conduct host-to-host or multihost audio teleconferences over an internet (multihost conferences require that the kernel support IP multicast). No special hardware other than a microphone is required for `vat` - sound I/O is via a Sparcstation's built-in audio hardware

  Currently available audio formats are

  1. pcm 64Kb/s 8-bit mu-law encoded 8KHz PCM
  2. idvi 32Kb/s Intel DVI ADPCM
  3. gsm 16Kb/s GSM
  4. lpc1 18Kb/s Linear Predictive Coder
  5. lpc4 8Kb/s Linear Predictive Coder

- Whiteboard `Wb` is a X11-based tool for shared drawing surface using an X-window among a group of users in the conference mode. `Wb` can also be used to export, view and annotate arbitrary PostScript files. To include PostScript images in the `wb` conference, the X-server has to either support *Display PostScript* or `wb` has to be able to exec the public domain postscript renderer `GhostScript`

These programs were developed by Van Jacobson and his group at the Lawrence Berkeley Laboratory in the University of California, Berkeley

The development of `wb`, `vat`, and `sd` were supported by the Director, Office of Energy Research, Scientific Computing Staff, of the U S. Department of Energy under Contract No. DE-AC03-76SF00098.

For the demonstration, *gyan*, *transport* and *cs10* were configured to participate in multicast sessions for audio and whiteboard conference

The present implementation for IP multicast does not assume error free transmission of multicast datagrams and it is the responsibilty of the upper layers to recover the lost packets In the concluding part of this thesis reliabilty issues in IP multicast have been discussed.

Chapter 5

Conclusion

In this project, low level group communication features such as multicasting, multicast routing and group management have been implemented on the IITK Campus network. A Multicast Backbone has been set-up, to allow hosts on different subnets within the IITK LAN, to participate in applications such as conferencing. This implementation is based on the public domain software available freely over the Internet, and has been modified to the extent needed for configuring hosts, as multicast routers and for subnet routing. This setup was successfully demonstrated for audio and whiteboard applications.

In the next section, the scope for further work is discussed.

5.1 Scope for Further Work

As an immediate extension of this project, the development work on the enhancement of the IITK-IP routers for multicast packet forwarding has already begun.

One of the major drawbacks of this IP multicast implementation is that it provides only an unreliable datagram delivery mechanism due to the use of the RAW or UDP socket interfaces. To enhance this implementation for use in applications having error-free datagram delivery requirements, a reliable group communication interface will have to be built on top of the existing implementation of IP multicast. The reliability aspects in IP multicast have been investigated by Armstrong *et al* [21]. This RFC also details the IP multicast transport layer.

In the following section, a brief description of the various reliability characteristics in group communications are included.

Since this project has been initiated with the intention of providing conferencing on the Campus LAN at a future date, the concluding section of this thesis briefly outlines some of the issues involved in designing group communication features for real-time conferencing applications.

5.1.1 Reliable group communication

The designers of the IP multicast enhancements as standardised in [19] have chosen to use the unreliable datagram delivery mechanism, since the application for which this implementation was targeted was *Audio and Video Conferencing*, which does not have a strict error recovery requirements and packet losses can be tolerated to some extent Applications such as multi-media multi-source information services require the group communication service to be reliable Thus, a combination of reliability mechanisms such as sequence numbers, timeouts and retransmission schemes have to be provided by the applications As the next level of enhancement, some reliability mechanism could be implemented in the upper layer to provide the use of IP multicast to applications requiring reliable group communications

H.Garcia and A Spauster [12] have discussed reliable group communications in detail and designed a protocol whose reliability can be evaluated based on a reliability criteria developed by them A gist of the reliability characteristics as outlined by them are reproduced here to highlight the reliabilty aspects in group communications.

The various reliability characteristics are as follows

- guarantee delivery of multicast datagram to group members based on the initiation time from the source
- Guarantee relative delivery delay of a message to group members This means that, once a member receives a message, the other members are guaranteed to receive the message within some maximum delay.
- Guarantee ordered delivery of multicast packets.
- Force sites to roll back and deliver messages that have been lost or inconsistently delivered to some or all members

5.1.2 Group communications for real-time conferencing

When designing the IP multicast, the IETF aimed at providing *loosely controlled teleconferences* where the attendees simply *tune in* to an agreed-upon multicast address and begin transmitting and/or receiving data. There is no co-ordination between end-systems and the conference state is constructed asynchronously through passive receipt of control messages from other group members. In conferencing applications, group co-ordination is an important part of the conference setup and control. The group-cordination protocol would essentially

- negotiate a common set of capabilities between the participants

- request participation
- initiate media connections
- propogate information among peers, and
- provide state synchronization

Inorder to provide such a session layer protocol for group co-ordination, at the transport layer, a *group messaging service* which provides *efficient transactions* is a must

The Multimedia Conferencing project [11] has focused on the networking requirements and towards this end, have designed and implemented a suite of experimental packet protocols that operate at a number of levels in the protocol stack As a next step in the direction of the development of multimedia conference control protocol, the development of a transport level [21] group messaging service for group communications could be taken up.

Appendix A

A.1 The IGMP overview :

Conceptually the IGMP is divided into two phases.

- Phase 1 When a host joins a new multicast group, it sends an IGMP message to the predefined "all hosts group address" declaring its membership The local multicast gateway receives the message and establishes the necessary routing information
- Phase 2 Because the membership may be dynamic, local multicast gateways periodically poll the hosts on the local netwrok to determine which hosts remain members. If no hosts report membership after the poll, the gateway stops advertising the group to other multicast routers, assuming that no hosts on the network are members of that group IGMP is an asymmetric protocol.

A.1.1 The IGMP implementation

All IGMP communication between hosts and multicast gateways use IP multicast. That is, when IGMP messages are encapsulated in an IP datagram for transmission, the IP destination address is the all_hosts_multicast_address This has been done to avoid congesting a local network Thus, datagrams carrying IGMP messgaes are hardware multicast if available. Also, a multicast gateway will not send individual request messages for each multicast groups. Instead, it sends a single poll message to request membership in all groups. The polling rate is restricted to atmost one request per minute. Hosts that are members of multiple groups do not send multiple responses at the same time but wait for a random time between 0 to 10 secs . During this time the hosts listen to responses from other hosts and supress any responses that are unnecessary Gateways do not keep an exact record of the group membership

and need to know only whether atleast one host on the local network remains a member of the group By spacing the responses and listening to other hosts responses , effectively only one host each group responds to a request message from a multicast gateway. This method has been used to reduce all unnecessary transmissions on the network.

A.1.2 The IGMP State Transition Diagram :

The IGMP must remember the status of each multicast group to which the host belongs. The host keeps a table in which the group membership information is recorded Initially, all entries are unused Whenever, an application program on the host joins a new group, IGMP software allocates an entry and fills in the information of the group Among the information, IGMP keeps a group reference counter which it initializes to 1 If any other application program also joins that group the counter is incremented As the application program drops out from the group, thr group refernce counter decrements The host leaves the group when the counter reaches zero

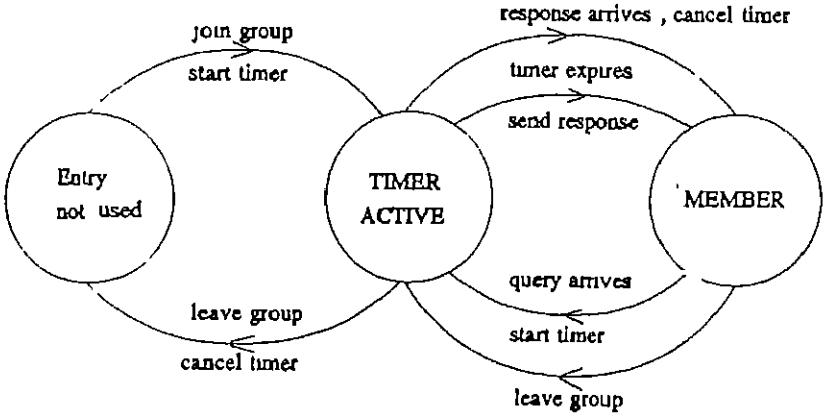


Figure A 1· IGMP State Transition Diagram

The action IGMP takes in response to IGMP messages can be best understood by the state transition diagram in Figure A 1. As the figure shows, a single timer mechanism can be used to generate both the initial response message as well as responses to request from multicast gateways. A request to join the group places the entry in the TIMER ACTIVE state and sets the timer to a small value. When the timer expires, IGMP generates and sends a response message and moves the entry to the MEMBER state. In the MEMBER state,

reception of an IGMP query causes the software to choose a timeout value, start a timer entry, and move the entry to the TIMER ACTIVE state If another host sends a response for the multicast group before the timer expires, IGMP cancels the timer and moves the entry back to the MEMBER state

A.2 The IGMP message format :

The IGMP message format is as shown in the Figure A 2

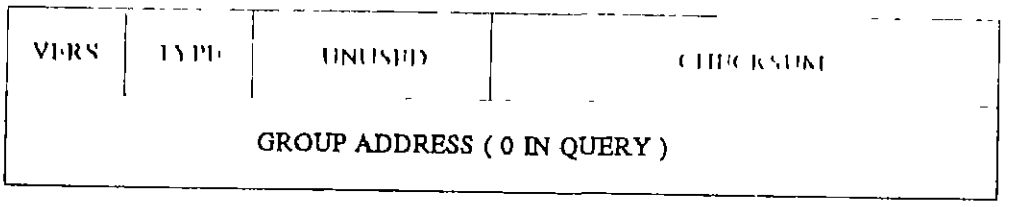


Figure A.2 IGMP Message Format

1. VERS = the protocol version (currently 1)
2. TYPE = 1 query sent by the multicast gateway;
   TYPE = 2 : response

   The Distance Vector Multicast Routing Protocol which is an experimental protocol for routing multicast packets, defines additional TYPE (3) for use in DVMRP messages.
3. UNUSED = zero
4. CHECKSUM field contains a checksum for the 8-octet IGMP message
5. GROUP ADDRESS = address of the group being reported in response message.

The IGMP is used to carry routing messages between routers

Appendix B

B.1 The DVMRP Commands

A DVMRP command consists of an 8-bit command numeric code, with atleast 8-bit data portion The commands as specified in RFC1075 [18] are reproduced below :

- NULL Command · 0

  Data Unused

  Description: The NULL command can be used to provide additional alignment or padding to 32 bits.

- Address Family Indicator (AFI) Command AFI Command 02

  Data· Family

  Values for family·

  2 = IP address family, in which addresses are 32 bits long

  Default Family = 2

  Description: The AFI command provides the address family for subsequent addresses in the stream (until a different AFI command is given). It is an error if the receiver does not support the address family.

- Subnetmask Command 03

  Data

  1. count · 0 or 1
  2. Additional argument, with AFI = IP: Subnetmask 4 bytes

  Default· Assume that following routes are to networks, and use a mask of the network mask of each route's destination.

Description: The Subnetmask command provides the subnet mask to use for subsequent routes. There are some requirements on the bits in the subnetmask: bits 0 through 7 must be 1, and all of the bits must not be 1.

If the count is 0, then no subnet mask applies, assume that the following routes are to networks, and use a mask of the network mask of each route's destination. If count is 1, then a subnet mask should be in the data stream, of an appropriate size given the address family.

It is an error for count not to equal 0 or 1.

Subnetmasks should not be sent outside of the appropriate network.

- Metric Command: 04

  Data: Value

  Value is the metric, as an unsigned value ranging from 1 to 255.

  Default: None.

  Description: The metric command provides the metric to subsequent destinations. The metric is relative to the router that sent this DVMRP routing update.

  It is an error for metric to equal 0.

- Flags0 Command: 05

  Data: Value. Meaning of bits in the value.

  Bit 7: Destination is unreachable. Bit 6: Split Horizon concealed route.

  Default: All bits zero.

  Description: The flags0 command provides a way to set a number of flags. The only defined flags, bits 6 and 7, can be used to provide more information about a route with a metric of infinity. A router that receives a flag that it does not support should ignore the flag.

  The command is called flags0 to permit the definition of additional flag commands in the future (flags1, etc.). This is an experimental command, and may be changed in the future.

- Infinity Command: 06

  Data: Value

  Value is the infinity, as an unsigned value ranging from 1 to 255.

  Default: Value = 16.

  Description: The infinity command defines the infinity for subsequent metrics in the stream.

It is an error for infinity to be zero, or less than the current metric

Destination Address (DA) Command . 07

Data Count

Array of 'count' additional arguments, with AFI = IP Destination Adresses

Count is the number of addresses supplied, from 1 to 255. The length of the addresses depends upon the current address family The number of addresses supplied is subject to the message length limitation of 512 bytes.

Default None.

Description: The DA command provides a list of destinations While this format can express routes to hosts, the routing algorithm only supports network and subnetwork routing The current metric, infinity, flags0 and subnetmask, when combined with a single destination address, define a route The current metric must be less than or equal to the current infinity

It is an error for count to equal 0.

Requested Destination Address (RDA) Command 08

Data Count

Array of 'count' additional arguments, with AFI = IP. Destination Addresses.

Count is the number of addresses supplied, from 0 to 255 The length of the addresses depends upon the current address family The number of addresses supplied is subject to the message length limitation of 512 bytes

Default: None.

Description: The RDA command provides a list of destinations for whom routes are requested. A routing request for all routes is encoded by using a count = 0

Non Membership Report (NMR) Command 09

Data · Count

Array of 'count' additional arguments, with AFI = IP. Multicast addresses, followed by the hold-down timer value (4 bytes each).

Count is the number of Multicast Address and Hold Down Time pairs supplied, from 1 to 255. The length of the addresses depends upon the

current address family. The number of pairs supplied is subject to the message length limitation of 512 bytes.

Default None

Description: The NMR command is experimental, and has not been tested in an implementation. Each multicast address and hold down time pair is called a non-membership report. The non-membership report tells the receiving router that the sending router has no descendent group members in the given group. Based on this information the receiving router can stop forwarding datagrams to the sending router for the particular multicast address(es) listed The hold down time indicates, in seconds, how long the NMR is valid.

It is an error for count to equal 0 The only other commands in a message that has NMR commands can be the AFI, flags0, and NULL commands. No relevant flags for the flags0 command are currently defined, but that may change in the future.

- Non Membership Report Cancel (NMR Cancel) Command · 10

  Data · Count

  Array of 'count' additional arguments, with AFI = IP: Multicast addresses

  Count is the number of Multicast Addresses supplied, from 1 to 255 The length of the addresses depends upon the current address family. The number of addresses supplied is subject to the message length limitation of 512 bytes

  Default: None.

  Description: The NMR Cancel command is experimental, and has not been tested in an implementation. For each multicast address listed, any previous corresponding non-membership reports are canceled. When there is no corresponding non-membership report for a given multicast address, the Cancel command should be ignored for that multicast address

  It is an error for count to equal 0. The only other commands in a message that has NMR Cancel commands can be the AFI, flags0, and NULL commands. No relevant flags for the flags0 command are currently defined, but that may change in the future.

An example :

Supplying a route to the IP addresses 128 2 251.231 and 128.2 236.2 with a metric of 2, an infinity of 16, a subnetmask of 255 255.255.0·

Subtype 1,

AFI 2, Metric 2, Infinity 16, Subnet Mask 255.255 255.0
[2] [2] [4] [2] [6] [16] [3] [1] [255] [255] [255] [0]
DA Count=2 [128.2.251.231] [128.2.236.2]
[7] [1] [128] [2] [251] [231] [128] [2] [236] [2]

B.2 Route Report Processing

For each route expressed in a routing message, the following must occur:

```
IF a metric was given for the route:
THEN add in the metric of the virtual interface
     that the message arrived on.

Lookup the route's destination address in the
routing tables.

IF the route doesn't exist in the tables:
THEN try to find a route to the same
     network in the routing tables.

IF that route exists in the tables:
   THEN IF this route came from the
           same router as the router
           that the found route came from:
        THEN CONTINUE with next route.

   IF route doesn't have a metric of infinity:
      THEN add the route to the routing tables.

   CONTINUE with next route.

IF this route came from the same router as
   the router that the found route came from:
THEN clear the route timer.

     IF a metric was received, and it is
        different than the found route's metric:
     THEN change the found route to use the new
          metric and infinity.
          IF the metric is equal to the infinity:
```

```
            THEN set the route timer to the
                 EXPIRATION_TIMEOUT.

            CONTINUE with next route.

        IF the received infinity does not equal the
           found route's infinity:
        THEN change the found route's infinity to be
             the received infinity.
             change the found route's metric to be
             the minimum of the received infinity and
             the found route's metric.

    ELSE IF a metric was received, and (it is less than the
            found route's metric or (the route timer is
            atleast halfway to the EXPIRATION_TIMEOUT and
            the found route's metric equals the received
            metric, and the metric is less than the received
            infinity)):

        THEN change the routing tables to use the received
             route, clear the route timer.

    CONTINUE with the next route.
```

B.3 Algorithm For Manipulating Leaf and - Child Interfaces

The algorithm for manipulating the children and leaf lists in route entries is:
Upon router startup.
Create a route entry for each virtual interface, with·

- all other virtual interfaces in its child list,
- an empty leaf list,
- no dominant router addresses, and
- no subordinate router addresses.

Start a hold down timer for each v-irtual inter face, with a value of LEAF_-TIMEOUT

Upon receiving a new route
Create the route entry, with:

- all virtual interfaces, other than the one on which the new route was received, in its child list,
- empty leaf list,
- no dominant router addresses, and
- no subordinate router addresses

Start the hold down timer for all virtual interfaces, other than the one on which the new route was received, with a value of LEAF_TIMEOUT.

Upon receiving a route on virtual interface V from neighbor N with a lower metric than the one in the routing table (or the same metric as the one in the routing table, if N's address is less than my address for V), for that route:

```
IF V is in the child list,
   delete V from the child list.

IF there is no dominant router for V and
   IF V is not (now) the next-hop virtual
   interface, record N as the dominant router.
```

Upon receiving a route on virtual interface V from neighbor N with a larger metric than the one in the routing table (or the same metric as the one in the routing table, if N's address is greater than my address for V), for that route:

```
IF N is the dominant router for V,
   delete N as the dominant router
   and add V to the child list.
```

Upon receiving a route from neighbor N on virtual interface V with a metric equal to infinity (the split horizon flag should also be set), for that route

```
IF V is in the leaf list,
   delete V from the leaf list.

IF there is no subordinate router for V,
   record N as the subordinate router.
```

Upon receiving a route from neighbor N on virtual interface V with a metric other than infinity (and no split horizon flag), for that route:

```
IF N is the subordinate router for V,
   delete N as the subordinate router and
   start the hold down timer for V.
```

Upon timer expiration for a virtual interface (V), for each route.

```
If there is no subordinate router for V,
add V to the leaf list.

Upon failure of neighbor N on virtual interface V,
for each route.

If N is the dominant router for V, delete N as the
dominant route and add V to the child list.

If N is the subordinate router for V, delete N as
the subordinate router and start the hold down
timer for V.
```

B.4 Timer Values

This section contains a list of the various rates and timeouts, their meanings, and their values. All values are in seconds.

How dynamic the routing environment is effects the following rates. A lower rate will allow quicker adaptation to a change in the environment, at the cost of wasting network bandwidth

FULL_UPDATE_RATE = 60
- How often routing messages containing complete routing tables are sent.

TRIGGERED_UPDATE_RATE = 5
- How often triggered routing messages may be sent out.

Raising the following rates and timeouts may increase the time that packets may be forwarded to a virtual interface unnecessarily

QUERY_RATE = 120

- How often local group membership is queried.

MEMBERSHIP_TIMEOUT = 2 * QUERY_RATE + 20

- How long a local group membership is valid without confirmation.

LEAF_TIMEOUT = 2 * FULL_UPDATE_RATE + 5

- How long the hold down timer is for a virtual interface.

Increasing the following timeouts will increase the stability of the routing algorithm, at the cost of slower reactions to changes in the routing environment.

NEIGHBOR_TIMEOUT = 4 * FULL_UPDATE_RATE

- How long a neighbor is considered up without confirmation. This is important for timing out routes, and for setting the children and leaf flags.

EXPIRATION_TIMEOUT = 2 * FULL_UPDATE_RATE

- How long a route is considered valid without confirmation. When this timeout expires, packets will no longer be forwarded on the route, and routing updates will consider this route to have a metric of infinity

GARBAGE_TIMEOUT = 4 * FULL_UPDATE_RATE

- How long a route exists without confirmation When this timeout expires, routing updates will no longer contain any information on this route, and the route will be deleted

Appendix C

C.1 Network Subsystem

In the UNIX operating system, network facilities are accessed through the "socket" abstraction for data transfer. A network subsystem provided by the UNIX network environment provides facilities such as :

- A structured interface to the socket level, that allows the development of network-independent application software.
- A consistent interface to the hardware devices used to transmit and receive data
- Network independent support for message routing
- Memory management.

The detailed description of the above is given in the 4 3BSD Unix Operating system manual [13].

C.1.1 Internal structure

Internally, the network subsystem is logically divided into three layers to manage the following tasks :
Interprocess data transport
Internetwork addressing and message routing
Transmission media support.

The first two layers are made up of modules that implement communication protocol; the software in the third layer is structurally much like a device driver. In the DARPA TCP/IP networking architecture these functions are handled by the three layers as described in [7]. The Figure C.1 illustrates the data flow across the layers in a network subsystem. The data, flows

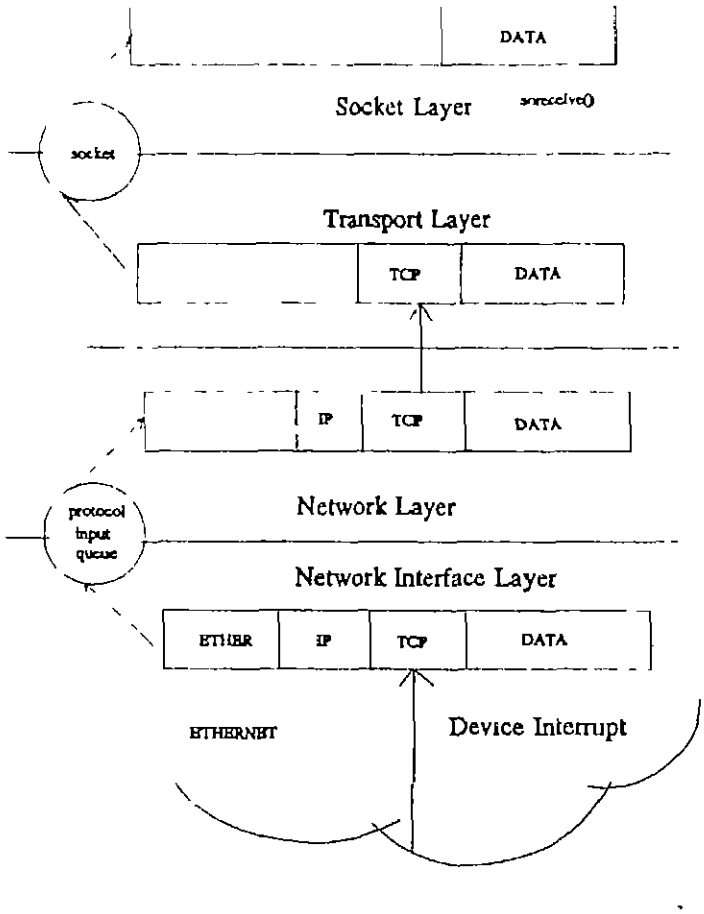


Figure C 1. Data Flow across the layers in a Network Subsystem

down to the network subsystem from the socket layer, through system calls to the transport layer modules that support the socket abstraction. The upward flow of data is asynchronously received, and passed from the network interface layer, to the appropriate communication protocol through per-protocol input message queues. "Software Interrupts" are used to schedule asynchronous network activity. The system schedules network protocol processing from the network-interface layer by marking a bit assigned to the protocol in the systems' network interrupt status word and posts a "software interrupt" reserved for triggering network activity The received data passes upwards through the communication protocols until it is placed in the receive queue of the destination socket. Each communication protocol module, for e g the DARPA TCP/IP, is made up of a collection of procedures and private data structures

Protocols are described by a protocol switch structure C.2 that contains the set of externally visible entry points and attributes. The socket layer

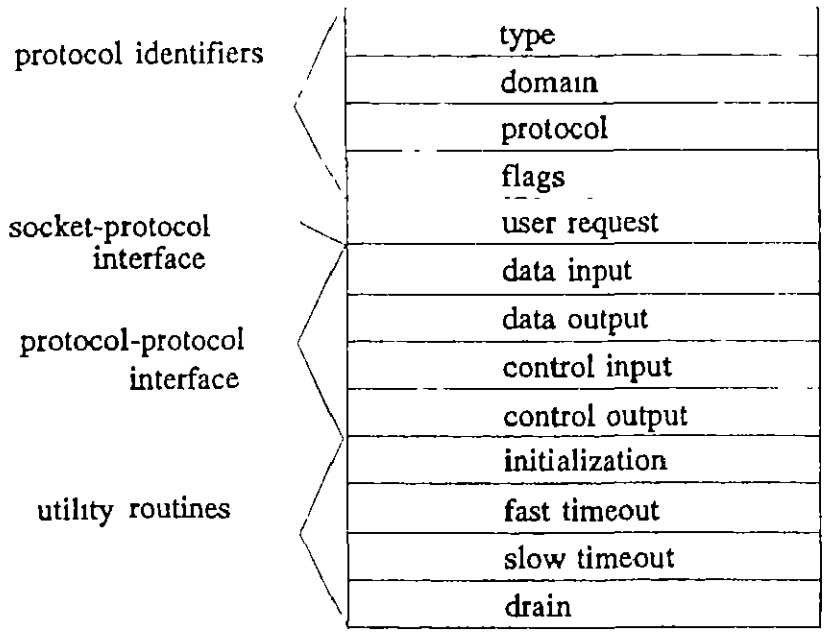


Figure C 2 Protocol Switch Architecture

interacts with a communication protocol only through the later's protocol switch structure, recording the structure's address in the socket's so_proto field. For more details refer Stevens : Unix Network Programming [24]

A network interface configured in a system, manipulates the hardware device and is responsible for encapsulation and decapsulation of any link-layer protocol header required to deliver a message to its destination. The selection of the network interface to use in delivering a packet, is a routing decision taken at the network layer of the particular protocol (e g IP layer in the TCP/IP protocol suite. A particular interface may have addresses in one or more address families. The interface addresses are setup at boot time using "ioctl" system call on a socket, in the appropriate domain The interface and its addresses are stored in a structure.

Bibliography

[1] S. R. Ahuja, J R Enson and D. Horn : 'The RAPPORT multimedia conferencing system', Proc Conference on Office Information Systems, Palo Alto, CA, March'88

[2] S R. Ahuja, J R Enson, and D D. Seligmann User Interfaces for Multimedia Multiparty Communications, Proc IEEE Infocom, (1165-1171), '93.

[3] Ariel J Frank, Larry D. Wittie and Arthur J. Bernstein Multicast Communication on Network Computers, IEEE Software Magazine, (49-60) May'85

[4] Bertsekas and Gallager . Data Networks, Prentice-Hall, '92.

[5] CCITT Recommendations H.261 for video coding and transmission at the rate of p*64 Kbps.

[6] Steve Casner : Most Frequently Asked Questions on MBONE, Jan'93 Available on the Anonymous ftp site : gregorio stanford.edu.

[7] Douglas Comer and David Stevens : Internetworking with TCP/IP, Vol 1. Prentice Hall '91.

[8] Y. K Dalal and R. M. Metcalfe : Reverse Path Forwarding of Broadcast packets, Communications of the ACM. 21(12) (1040-1048), Dec'78.

[9] S. Deering . Multicast Routing in Internetworks and Extended LANs, SIGCOMM Summer '88 Proceedings, Aug'88.

[10] S Deering and D. R. Cheriton : Multicast Routing in Datagram Internetworks and Extended LANs, ACM Trans. On Computer Systems 8(2), (85-110), May'90

[11] Eve M Schooler : Case Study: Multimedia Conference Control in a Packet-switched Teleconferencing System. Internetworking Research and Experience , Vol. 4, (99-120), '93.

[12] Garcia-Molina and Annemarie Spauster : Ordered and Reliable Multicast Communications, ACM Trans On Computer Systems, Vol 9 No 3, (242-271), Aug'91

[13] Samuel Leffler, Marshall Kirk McKusick, Micheal Karels, John Quarterman . The Design and Implementation of 4.3BSD UNIX operating system. Addison - Wesley Publishing Company, '89

[14] J. Postel : DARPA standard for Internet Protocol, RFC791[1] May'81

[15] J. Postel 'Transport Control Protocol', RFC 793

[16] S Deering : Host Extensions for IP Multicasting, RFC988, Aug'88 (obsoleted by RFC1054 and subsequently by RFC1112).

[17] C. Hedrick Routing Information Protocol· RFC 1058, July'88

[18] D Waitzman, C Partridge, S. Deering : Distance Vector Multicasting Protocol· RFC 1075, Nov'88

[19] S. Deering . Host Extensions for IP Multicasting, RFC1112, June'90.

[20] John Moy . Open Shortest Path First (OSPF), RFC1131, May'92.

[21] S Armstrong, A Frier, and K. Marzullo : Multicasting Transport Protocol, RFC1301, '92.

[22] Shiio Sakata: Development and evaluation of an In-house Multimedia Desktop Conference System. IEEE JSAC, Vol 8 No. 3, (340-347), Apr'90

[23] H Schulzrinne . 'First IETF Internet Audiocast', ACM SIGCOMM Computer Communications Review, (3), (92-97), '92.

[24] W Richard Stevens: Unix Network Programming, Prentice Hall of India, '93.

[25] Wu-Hon, F. Leung, Thomas J. Baumgarter : A Software Architecture for Workstations Supporting Multimedia Conferencing in Packet Switching Networks, IEEE JSAC, Vol. 8 No. 3, (380-390), Apr'90

[1]The RFCs referred, have been obtained from the anonymous ftp server in IITK.

EE-1994-M-SUL-IMP.